UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Charbel Szymanski

DESENVOLVIMENTO DE TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS PARA INSPEÇÃO DE PLACAS DE CIRCUITO IMPRESSO PRODUZIDAS EM PEQUENAS SÉRIES

Florianópolis

2014

Charbel Szymanski

DESENVOLVIMENTO DE TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS PARA INSPEÇÃO DE PLACAS DE CIRCUITO IMPRESSO PRODUZIDAS EM PEQUENAS SÉRIES

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas.
Orientador: Prof. Marcelo Ricardo Stemmer, Dr.

Florianópolis

2014

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

Szymanski, Charbel
Desenvolvimento de técnicas de processamento digital de imagens para inspeção de placas de circuito impresso produzidas em pequenas séries / Charbel Szymanski ; orientador, Marcelo Ricardo Stemmer - Florianópolis, SC, 2014
116 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de Automação e Sistemas.
2. processamento de imagens. 3. inspeção óptica automática.
4. produção de pequenas séries. 5. filtragem de falsos matches.
I. Stemmer, Marcelo Ricardo. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

Charbel Szymanski

DESENVOLVIMENTO DE TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS PARA INSPEÇÃO DE PLACAS DE CIRCUITO IMPRESSO PRODUZIDAS EM PEQUENAS SÉRIES

Esta Dissertação foi julgada adequada para a obtenção do Título de "Mestre em Engenharia de Automação e Sistemas", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 27 de fevereiro de 2014.

Prof. Jomi Fred Hübner, Dr.
Coordenador do Curso

Prof. Marcelo Ricardo Stemmer, Dr.
Orientador

Banca Examinadora:

Prof. Marcelo Ricardo Stemmer, Dr.
Presidente

Prof. Marcos Marinovic Doro, Dr.
Instituto Federal de São Paulo

Prof. Armando Albertazzi Gonçalves Júnior, Dr.
LABMETRO/UFSC

Prof. Carlos Barros Montez, Dr.
DAS/UFSC

Este trabalho é dedicado a todos aqueles que fazem ciência para o bem da humanidade e livres de preconceitos materialistas.

AGRADECIMENTOS

Agradeço à minha família e a meus amigos pelo apoio durante todo o período do mestrado.

Agradeço também ao professor Marcelo Ricardo Stemmer, pela paciência e pelas orientações.

Aos meus colegas de mestrado, pelo companheirismo.

Ao Departamento de Automação e Sistemas da UFSC, pela oportunidade de realizar este trabalho.

À universidade RWTH-Aachen, pela oportunidade de realizar intercâmbio de pesquisa.

Ao CNPq e CAPES pelo seu financiamento.

Não tentes ser bem sucedido, tenta antes ser um homem de valor.

Albert Einstein

RESUMO

Há muito tempo existe a preocupação em aprimorar as técnicas de processamento de imagens para inspeção de PCI. No entanto, percebe-se que o foco está principalmente voltado à inspeção da produção em massa. Mesmo com um volume de produção bem inferior ao de uma produção em massa, a produção em pequenas séries tem se tornado cada vez mais relevante. Neste trabalho é apresentado o estado da arte da inspeção de PCI, não necessariamente voltada à inspeção de pequenas séries. Diferentes técnicas de processamento de imagens e visão computacional foram pesquisadas na literatura relacionada, visando desenvolver uma arquitetura de processamento de imagens que fosse adequada à inspeção de pequenas séries. Dentre as técnicas encontradas, percebeu-se que o algoritmo SIFT possui um grande potencial para contornar os problemas que surgem na utilização de técnicas de inspeção de produção em massa, na produção de pequenas séries. Dessa forma, a arquitetura de processamento de imagens proposta e implementada neste trabalho, está fortemente baseada na utilização do algoritmo SIFT. Para que se obtivesse melhores resultados com o uso do SIFT, foram desenvolvidos dois algoritmos para filtragem de falsos *matches*. Um software de inspeção foi implementado, utilizando-se a arquitetura proposta. Na observância do resultado dos experimentos realizados com esse software, percebe-se que a arquitetura é adequada à produção de pequenas séries, possibilitando uma taxa de acertos acima de 80%, em um ambiente real.

Palavras-chave: Processamento de imagens, Inspeção óptica automática, Produção de pequenas séries, Filtragem de falsos matches, Placas de circuito impresso

ABSTRACT

There has long been a concern to improve the techniques of image processing for PCB's inspection. However, it is noticed that the focus is mainly on the inspection of mass production. Even with a volume of production well below of a mass production's volume, small series production has become increasingly relevant. In this paper is presented the state of the art of PCB's inspection, not necessarily focused on the inspection of small series. Different techniques of image processing and computer vision were researched in the related literature in order to develop an architecture for image processing that is suitable to the inspection of small series. Among the techniques found, it was realized that the SIFT algorithm has a great potential to overcome the problems that arise in the use of inspection techniques of mass production, in the production of small series. Thus, the architecture for image processing proposed and implemented in this work is strongly based on the use of the SIFT algorithm. In order to obtain best results using the SIFT, two algorithms for filtering false matches have been developed. An inspection software was implemented using the proposed architecture. Subject to the results of experiments performed with this software, it is noticed that the architecture is suitable for production of small series, allowing an accuracy ratio above 80%, in a real environment.

Keywords: image processing, automated optical inspection, small series production, false matches filtering, printed circuit board

LISTA DE FIGURAS

Figura 1 Estratégia para possibilitar aos sistemas de produção um comportamento auto-otimizável. Fonte: (STEMMER et al., 2011)... 26
Figura 2 Sistema de visão computacional S2iAOI 27
Figura 3 Linha de montagem SMT do LABelectron. Fonte: (MELO, 2013)..... 29
Figura 4 Foto de um ambiente de escritório, com círculos representando a localização dos *keypoints* gerados..... 43
Figura 5 Subtração de gaussianas nas diferentes oitavas do espaço de escala. Fonte: (LOWE, 2004)..... 45
Figura 6 Detecção de máximos e mínimos na vizinhança de 8 de um determinado *pixel*, marcado com X. Fonte: (LOWE, 2004)..... 46
Figura 7 Representação de um possível histograma utilizado para determinar a orientação de um *keypoint*. Fonte: (SINHA, 2013).... 49
Figura 8 Representação da janela de amostras computada ao redor do *keypoint* para formar o descritor. Fonte: Adaptado de (SINHA, 2013)..... 50
Figura 9 Fluxograma da arquitetura de processamento de imagens proposta..... 59
Figura 10 Exemplo de componentes eletrônicos com e sem serigrafia..... 60
Figura 11 *Keypoints* gerados a partir de fotos de componentes eletrônicos SMD sem serigrafia..... 61
Figura 12 Fluxograma representativo do processo que verifica se uma determinada imagem se refere ao fundo da placa ou a algum componente 66
Figura 13 Ocorrência de falsos matches 74
Figura 14 Fotos de um ambiente de escritório, utilizadas nos testes dos filtros desenvolvidos 77
Figura 15 Matching entre as duas fotos apresentadas na figura 14 77
Figura 16 Resultado da filtragem preliminar dos *matches* da figura 15, baseada apenas na menor distância euclidiana entre os *matches* 78
Figura 17 Resultado da aplicação do filtro por coerência de rotação nos *matches* apresentados na figura 16..... 84
Figura 18 *Matches* eliminados no filtro por coerência de rotação.. 84

Figura 19 Resultado da aplicação do filtro por coerência de translação nos *matches* apresentados na figura 16 87
Figura 20 *Matches* eliminados no filtro por coerência de translação 88
Figura 21 *Matching* de fotos reais de dois componentes eletrônicos do mesmo tipo 89
Figura 22 Resultado da aplicação do filtro por coerência de rotação nos *matches* apresentados na figura 21 90
Figura 23 Resultado da aplicação do filtro por coerência de translação nos *matches* apresentados na figura 21 90
Figura 24 *Matches* eliminados na aplicação do filtro por coerência de rotação aos *matches* apresentados na figura 21 91
Figura 25 *Matches* eliminados na aplicação do filtro por coerência de translação aos *matches* apresentados na figura 21 91
Figura 26 Vista lateral do sistema de iluminação e câmera, utilizados na S2iAOI 94
Figura 27 Vista de baixo do sistema de iluminação e câmera, utilizados na S2iAOI 94
Figura 28 Resultado do matching entre um componente e uma grande área do fundo da placa 97
Figura 29 Componentes reconhecidos como sendo os mesmos, em função de boa parte da serigrafia ser semelhante. 100
Figura 30 Matching entre fotos de componentes com partes repetidas na serigrafia 100
Figura 31 Resultado do matching utilizando fotos de componentes não-SMD 102
Figura 32 Resultado do matching utilizando fotos de componentes com superfície reflexiva 103

LISTA DE TABELAS

Tabela 1 Distribuição de frequências das faixas de rotação 83
Tabela 2 Contabilização dos resultados das inspeções para os defeitos existentes e os encontrados 96
Tabela 3 Resumo do desempenho e defeitos não encontrados nas inspeções 97

LISTA DE ABREVIATURAS E SIGLAS

SMD	Surface-Mount Device	25
PPS	Produção de Pequenas Séries	26
BRAGECRIM	Brazilian German Collaborative Research Initiative on Manufacturing Technology	26
S2i	Grupo de pesquisa em Sistemas Industriais Inteligentes	26
DAS	Departamento de Automação e Sistemas	26
UFSC	Universidade Federal de Santa Catarina	26
WZL	Werkzeugmaschinenlabor	26
RWTH-Aachen	Rheinisch-Westfaelische Technische Hochschule - Aachen University	26
COGMET	Cognitive Metrology for Flexible Small Series Production	26
S2iAOI	Automated Optical Inspection machine from Industrial Intelligent Systems group	27
PDI	Processamento Digital de Imagens	27
PCI	Placa de Circuito Impresso	27
THT	Through-Hole Technology	29
SMT	Surface-Mount Technology	29
SPI	Solder Paste Inspection	29
PPL	Produção em Pequenos Lotes	31
CCD	Charge-Coupled Device	34
CI	Circuito Integrado	35
MAD	Mean Absolute Difference	38
MSE	Mean Square Difference	38
SAD	Sum of Absolute Differences	42
SSD	Sum of Squared Differences	42
PCA	Principal Components Analysis	53
OCR	Optical Character Recognition	65
OCV	Optical Character Verification	65

SUMÁRIO

1 INTRODUÇÃO 23
1.1 OBJETIVOS DO TRABALHO 24
1.1.1 Objetivo geral 24
1.1.2 Objetivos específicos 25
1.2 CONTEXTUALIZAÇÃO DO TRABALHO 25
1.3 ORGANIZAÇÃO DA DISSERTAÇÃO 29
2 FUNDAMENTAÇÃO TEÓRICA 31
2.1 GARANTIA DE QUALIDADE EM PPS 31
2.2 INSPEÇÃO DE PCI 32
2.2.1 Inspeção de placa nua 33
2.2.2 Inspeção de placa montada 33
2.3 METROLOGIA ÓPTICA 34
2.4 TRABALHOS RELACIONADOS 35
2.5 TEMPLATE MATCHING 40
2.6 ALGORITMO SIFT 42
2.6.1 Detecção de extremos no espaço de escala 44
2.6.2 Localização e eliminação de keypoints 46
2.6.3 Atribuição de orientação aos keypoints 48
2.6.4 Geração dos descritores 50
2.6.5 Variações do SIFT 53
2.7 CONCLUSÃO 54
3 PROJETO E IMPLEMENTAÇÃO DE UM SISTEMA DE INSPEÇÃO VISUAL DE PCI 55
3.1 ANÁLISE DE REQUISITOS 56
3.2 ARQUITETURA DE PROCESSAMENTO DE IMAGENS 58
3.3 FERRAMENTAS E TÉCNICAS UTILIZADAS 69
3.4 ESTRUTURA DO SOFTWARE DESENVOLVIDO 70
3.5 CONCLUSÃO 72
4 ELIMINAÇÃO DE FALSOS MATCHES BASEADA NA COERÊNCIA DE ROTAÇÃO E TRANSLAÇÃO 73
4.1 FALSOS MATCHES 73
4.2 TRABALHOS RELACIONADOS 74
4.3 COERÊNCIA ENTRE MATCHES 76
4.4 COERÊNCIA DE ROTAÇÃO 77
4.5 COERÊNCIA DE TRANSLAÇÃO 83
4.6 PRECISÃO E LIMITAÇÕES 87
5 AVALIAÇÃO DOS RESULTADOS 93

6 CONSIDERAÇÕES FINAIS 105
6.1 PERSPECTIVAS PARA TRABALHOS FUTUROS 106
REFERÊNCIAS 109

1 INTRODUÇÃO

A produção de placas eletrônicas é um mercado em ascensão e de grande relevância na produção de equipamentos eletrônicos. Esse fator é impulsionado não somente pelo aumento na produção de equipamentos eletrônicos, como também, pelo surgimento de novos equipamentos. No ano 2000, o mercado mundial de placas eletrônicas chegou a US$ 42,7 bilhões (MELO; RIOS; GUTIERREZ, 2001). Em 2012, esse mercado atingiu a marca de US$ 60 bilhões (IPC, 2013a), ou seja, entre 2000 e 2012, registrou-se um crescimento de aproximadamente 40%. Somente em relação à montagem final das placas eletrônicas, o mercado mundial movimentou aproximadamente US$ 2 bilhões, em 2013 (IPC, 2013a). A tendência para 2017 é que o mercado mundial de produção de PCI chegue a US$ 93,9 bilhões (LUCINTEL, 2013).

Liderando esse mercado, a China representou em 2012, 42,8%, seguida pelo Japão, com 14,7%. Os EUA, que há alguns anos dominava esse mercado, representou em 2012, apenas 4,6% do mercado, ocupando o 5º lugar entre os maiores produtores mundiais (IPC, 2013b).

Não é somente a quantidade de placas produzidas que tem aumentado, a complexidade das mesmas também. Como consequência, esse aumento na complexidade das placas produzidas tornou ainda mais importante os mecanismos de controle de qualidade. Para complicar esse cenário, verifica-se que houve também uma redução no tempo de atualização dos produtos, bem como, o surgimento de produtos destinados a atender necessidades mais específicas (produtos mais personalizados). Isso aumentou a demanda da produção de pequenas séries (PPS).

Apesar da produção de poucas unidades, espera-se que a PPS ofereça o mesmo nível de qualidade da produção em massa. Com isso, surgem novos desafios no que se refere ao controle de qualidade. Uma das etapas fundamentais no controle da qualidade é a etapa de inspeção, que visa detectar os defeitos oriundos da montagem das placas. As inspeções são realizadas em diversos pontos na linha de produção, porém, em função do alto custo dos equipamentos de inspeção ou do baixo custo das placas produzidas, muitas vezes essa etapa é alocada no final do processo. Especialmente na PPS, verifica-se que a inspeção feita por pessoas ainda é realizada.

Para que se possa ter garantia de qualidade na montagem de placas, é necessário realizar o *setup* adequadamente, conforme o modelo da placa sendo inspecionada. Entretanto, em uma PPS, muitas vezes o

tempo de *setup* torna-se extenso, considerando o tempo de produção, ou seja, o tempo de setup acaba se tornando proporcionalmente muito maior do que o tempo de *setup* para uma produção em massa. Com isso, muitas vezes o operador acaba por realizar um *setup* mais grosseiro, dando margem à ocorrência de falhas na inspeção. Dessa forma, a inspeção realizada por seres humanos acaba sendo aplicada no processo de montagem. O problema da inspeção feita por pessoas é justamente a existência dos fatores inerentes ao seres humanos como, por exemplo, fadiga e tédio. Por esse motivo, a inspeção automática acaba sendo mais confiável do que a inspeção humana (DORO, 2009).

Nesse cenário, torna-se importante o aprimoramento dos métodos de inspeção automática, visando torná-los mais adequados à PPS. Dentre os métodos de inspeção automática mais flexíveis, destaca-se a inspeção óptica. Para entender a importância da inspeção óptica, ressalta-se que a maior parte das aplicações de visão computacional na indústria se trata de sistemas ópticos de inspeção (MALAMAS et al., 2003).

Para o desenvolvimento de um sistema de inspeção óptica, este trabalho faz uso de técnicas de processamento de imagens e visão computacional. A área de visão computacional tem crescido rapidamente, fato esse acarretado, principalmente, pelos seguintes fatores (BRADSKI; KAEHLER, 2008):

- Barateamento e aperfeiçoamento das câmeras;
- Barateamento dos dispositivos de processamento;
- Amadurecimento dos algoritmos utilizados em visão computacional.

Para se entender a aplitude da área de visão computacional, destaca-se que, praticamente em todos os tipos de produção em massa utiliza-se alguma técnica de visão computacional, para se realizar inspeção automática (BRADSKI; KAEHLER, 2008).

1.1 OBJETIVOS DO TRABALHO

1.1.1 Objetivo geral

Este trabalho tem por objetivo a pesquisa e o desenvolvimento de técnicas de processamento digital de imagens, para garantia da qualidade em linhas de produção de PCI em pequenas séries. No contexto

da produção de PCI, este trabalho está focado na inspeção óptica automática de PCI montadas. Dentre as possíveis placas a serem inspecionadas, este trabalho destina-se a inspecionar apenas placas baseadas em componentes com algum tipo de serigrafia ou marcação de identificação na superfície.

1.1.2 Objetivos específicos

Para atingir o objetivo geral do trabalho, foram traçados os seguintes objetivos específicos:

- Elaborar uma arquitetura de processamento de imagens que possibilite inspeções automáticas confiáveis e ao mesmo tempo, seja adequada à PPS;
- Implementar um software de inspeção, baseado na arquitetura de processamento de imagens proposta;
- Validar o software implementado, utilizando fotos reais de PCI, adquiridas com a atual versão do sistema de visão computacional (S2iAOI), desenvolvido no projeto BRAGECRIM 013/09.

1.2 CONTEXTUALIZAÇÃO DO TRABALHO

Com o propósito de melhorar a eficiência da produção de produtos de alta qualidade em linhas flexíveis para produção de pequenas séries (PPS), foi criado um projeto no âmbito do programa BRAGECRIM (Brazilian German Collaborative Research Initiative on Manufacturing Technology), um convênio de cooperação científica entre Brasil e Alemanha para o desenvolvimento de pesquisas na área de manufatura. O projeto em questão, identificado como BRAGECRIM 013/09 (STEMMER et al., 2011), é uma parceria entre o grupo de pesquisa S2i/DAS/UFSC e o LABelectron/CERTI pelo lado brasileiro e o Instituto WZL/RWTH-Aachen pelo lado alemão, com a meta de "Desenvolver uma nova geração de sistemas metrológicos e de garantia da qualidade da produção com capacidades adaptativas e alto grau de percepção cognitiva em relação ao produto e ao processo" (PFEIFER et al., 2010). O projeto é intitulado COGMET (Cognitive Metrology for Flexible Small Series Production - Metrologia cognitiva para a produção flexível em pequenas series) e conta com o apoio da CAPES, FINEP e CNPq pelo lado brasileiro e do DFG pelo lado alemão. Neste projeto, o

problema da garantia da qualidade dos produtos e processos em linhas de PPS é abordado através do desenvolvimento dos chamados Sistemas Auto-Otimizáveis, que possuem as propriedades de flexibilidade e mutabilidade, autonomia e cognição, conforme ilustrado na figura 1.

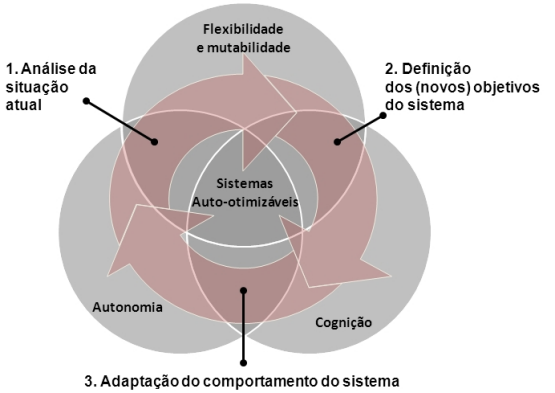


Figura 1 – Estratégia para possibilitar aos sistemas de produção um comportamento auto-otimizável. Fonte: (STEMMER et al., 2011).

Nesse cenário, ocorre a interação de 3 fatores:

1. **Análise da situação atual**: percepção do ambiente ao redor do sistema;

2. **Definição dos (novos) objetivos do sistema**: com base na situação atual, novos objetivos podem ser determinados pelo sistema, de forma autônoma;

3. **Adaptação do comportamento do sistema**: alteração dos parâmetros, estrutura e comportamento do sistema, de acordo com os novos objetivos estipulados.

O projeto propõe que a flexibilidade e a mutabilidade de um sistema de produção para pequenas séries seja alcançada com o emprego de tecnologias como: integração de sensores, fusão de dados e sistemas ópticos de medição, sendo esse último o foco deste trabalho.

Dentro do contexto apresentado, este trabalho está focado na pesquisa e desenvolvimento de técnicas de Processamento Digital de

Imagens (PDI) para um sistema de visão computacional flexível e reconfigurável, utilizado em linhas de produção de placas de circuito impresso (PCI) em pequenas séries. O sistema de visão mencionado já conta com um protótipo em desenvolvimento pelo S2i/DAS/UFSC no contexto do projeto mencionado anteriormente. Esse protótipo, apresentado na figura 2, denominado S2iAOI (Automated Optical Inspection machine from Industrial Intelligent Systems group) foi desenvolvido em uma dissertação de mestrado, que teve como foco o desenvolvimento do projeto mecatrônico e a implementação do software de controle e operação da máquina (MELO, 2013). As técnicas de PDI desenvolvidas neste trabalho foram elaboradas com o intuito de serem implantadas no S2iAOI.

Figura 2 – Sistema de visão computacional S2iAOI

Para garantir a autonomia do sistema, o projeto COGMET está sendo desenvolvido, tanto no lado brasileiro como no lado alemão, baseado no paradigma de sistemas multiagente (WOOLDRIDGE, 1999)(WOOLDRIDGE, 2002). Sendo assim, o S2iAOI está sendo desenvolvido de forma que possa funcionar como um agente de software visando sua inserção em um sistema multiagente que controlará toda a linha de produção de placas de circuito impresso. A implementação de um Sistema Auto-otimizável com o foco no paradigma multiagente, objetivando promover o controle da qualidade na PPS, com aplicação à

inspeção de placas de circuito impresso, está sendo desenvolvido como uma tese de doutorado, no contexto do projeto BRAGECRIM 013/09 (ROLOFF, 2013).

Sistemas multiagente são sistemas compostos por softwares, denominados agentes, que interagem entre si. A autonomia e a interação com outros agentes e o ambiente em que estão inseridos são as capacidades (funcionalidades) principais de um agente. Um agente de software é capaz de estipular seus próprios objetivos de forma independente e agir para atingi-los, tudo dentro dos limites para os quais foi construído (WOOLDRIDGE, 2002).

No lado brasileiro, o projeto BRAGECRIM 013/09 conta com o parceiro LABelectron, um laboratório-fábrica pertencente à fundação CERTI. Nesse laboratório são montadas placas de circuito impresso baseadas nas tecnologias THT e SMT. A figura 3 apresenta o fluxo da linha de montagem SMT do LABelectron.

A Printer é a máquina responsável pela aplicação da pasta de solda dos componentes SMD. Após a aplicação da pasta de solda, é necessário verificar se a mesma foi aplicada corretamente, o que é feito pela SPI (Solder Paste Inspection). Uma vez que a pasta de solda esteja aplicada corretamente, os componentes podem ser inseridos automaticamente pela insersora. No entanto, após a etapa de inserção automática, geralmente alguns componentes precisam ser inseridos manualmente. Isso ocorre pelo fato de não haver alimentador[1] disponível na máquina ou o lote ser muito pequeno, em que não é possível colocar componentes no alimentador. Estando todos os componentes eletrônicos inseridos, a placa pode passar pelo forno de refusão para que os componentes sejam efetivamente soldados na placa. No final da linha de montagem SMT do LABelectron há uma etapa de inspeção automática, para garantia de qualidade. Nessa etapa é utilizada uma máquina AOI (Automated Optical Inspection) para detectar possíveis defeitos ocorridos durante a montagem. Como a AOI está instalada no final da linha, caso seja detectado algum defeito, a placa poderá ser descartada, dependendo de sua finalidade, ou quando o retrabalho for mais caro do que a placa montada. Isso ocorre porque nesta etapa a placa já passou pelo forno de refusão. Sendo assim, o foco do desenvolvimento do S2iAOI neste trabalho é torná-lo apto para ser utilizado nessa linha de produção do LABelectron, em uma etapa anterior ao forno de refusão e após a inserção manual de componentes. Com isso, o sistema de inspeção estará destinado a verificar somente os defeitos de

[1] Alimentador é um suporte, utilizado pela insersora, no qual os componentes eletrônicos são fixados para serem posteriormente inseridos nas PCI.

inserção de componentes e não irá contemplar os defeitos de soldagem.

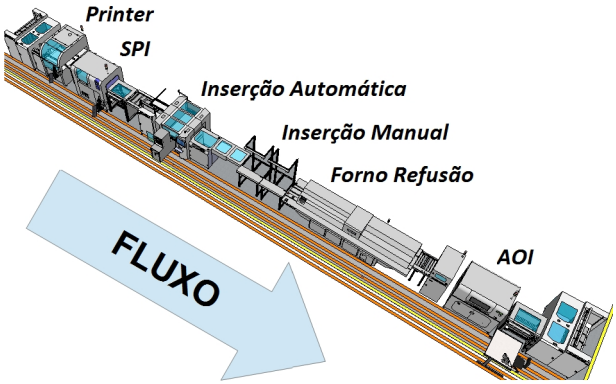


Figura 3 – Linha de montagem SMT do LABelectron. Fonte: (MELO, 2013).

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

O capítulo 2 contém uma visão geral dos fundamentos teóricos que embasaram o desenvolvimento desta dissertação. O sistema de inspeção desenvolvido, incluindo a arquitetura de processamento de imagens proposta, é apresentado no capítulo 3. No capítulo 4 são apresentados dois algoritmos criados para filtragem de falsos *matches*, cuja necessidade surgiu no decorrer do desenvolvimento do sistema de inspeção. Em função da relevância para esta dissertação, esses algoritmos são apresentados em um capítulo específico. A validação do software implementado é apresentada no capítulo 5, aonde são apresentados e avaliados os resultados dos experimentos realizados com o software implementado. No capítulo 6 são apresentadas as conclusões deste trabalho, bem como, sugestões para o desenvolvimento de trabalhos futuros, visando melhorar ou ampliar diversos aspectos do sistema de inspeção que foi desenvolvido nesta dissertação.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir são apresentados alguns conceitos relativos ao conteúdo deste trabalho. Por este trabalho tratar-se de uma solução para inspeção de PCI, no contexto da PPS, é apresentada uma introdução aos principais conceitos dessas áreas, bem como, uma introdução à metrologia óptica, em que se caracteriza o tipo de solução proposta neste trabalho. Como ponto de partida para o desenvolvimento da solução proposta neste trabalho, as soluções encontradas para o mesmo problema foram estudadas e são apresentadas no item 2.4. Posteriormente é apresentado o conceito de *template matching*, extensamente utilizado na literatura relacionada ao presente assunto. Por fim, é apresentado o algoritmo SIFT, que é utilizado como parte central do *pipeline* (arquitetura) de processamento de imagens, desenvolvido neste trabalho.

2.1 GARANTIA DE QUALIDADE EM PPS

As linhas de produção têm sofrido mudanças, seguindo uma tendência de personalização dos produtos. Isso faz com que as mesmas tenham de ser cada vez mais flexíveis, para atenderem às exigências do mercado, que frequentemente requer mudanças nas características e na quantidade dos produtos a serem produzidos. Nesse cenário, a PPS está cada vez mais em foco. Apesar de não haver uma definição única para o conceito de PPS, as seguintes características são comumente mencionadas pelos autores (JADHAV, 2005) (DORO, 2009) (SCHMITT; PAVIM, 2009):

- alta variedade de produtos produzidos;
- curto tempo de produção por lote;
- baixo volume de produção, sendo que, pode até ocorrer a produção de uma única unidade por lote.

A PPS, também conhecida como produção em pequenos lotes (PPL), está fortemente baseada na flexibilidade e mesmo assim, objetiva atingir os mesmos níveis de eficiência da produção em massa (DORO, 2009).

A aplicação de métodos estatísticos utilizada para garantia de qualidade na produção em massa perde aplicabilidade à PPS em função da baixa quantidade de produtos produzidos (SCHMITT; PAVIM, 2009) (DORO, 2009). A necessidade por flexibilidade na PPS resulta naquele

que, segundo (SCHMITT; PAVIM, 2009), é considerado o maior desafio da PPS: garantir a qualidade desde o primeiro lote.

Dentre os problemas relativos à garantia de qualidade em PPS, relacionados por (SCHMITT; PAVIM, 2009), destaca-se:

- falta de previsibilidade sobre o produto e o processo;
- elevado tempo de *setup* e nenhum ou apenas poucos produtos descartáveis a serem usados para ajustar os processos;
- curto tempo para observar e corrigir o processo durante a produção;
- falta de dados para tomada de decisão.

2.2 INSPEÇÃO DE PCI

Os defeitos gerados durante a produção de PCI podem ser ocasionados por diversos fatores relacionados ao produto e ao processo. Como não há uma técnica capaz de detectar todos os tipos de defeitos, o ideal é a combinação de diferentes técnicas (DORO, 2013).

Os sistemas de inspeção podem, de um modo geral, ser divididos em dois grandes grupos: com contato físico e sem contato físico. Especialmente no contexto deste trabalho, são abordadas técnicas que não utilizam contato físico entre o sistema de inspeção e a PCI. No caso de PCI, um exemplo de sistema de inspeção que usa contato físico são os métodos de teste elétrico de placas, nos quais são procurados defeitos como, por exemplo, rompimento de trilhas. Nesse caso, o teste elétrico é utilizado para verificar a continuidade das trilhas impressas.

A inspeção de PCI, produzidas em THT ou SMT, pode ser dividida em duas categorias: inspeção de placa nua e inspeção de placa montada. Apesar de um sistema de inspeção poder envolver ambas as categorias de inspeção, normalmente um mesmo sistema trata de somente uma delas. Isso pode ser entendido pelo fato de os tipos de defeitos procurados em cada caso serem bem diferentes, o que leva à utilização de técnicas diferentes de inspeção.

Pode-se ainda classificar a inspeção de PCI em:

- Baseada em referência: utiliza como referência algum modelo de placa sem defeito, de forma que, diferenças entre a imagem da placa de referência (sem defeito) e a imagem de uma placa de teste poderão ser consideradas defeitos;

- Não baseada em referência: a detecção de defeitos é baseada na comparação entre a imagem adquirida da placa de teste e o projeto da placa;

- Híbrida: a inspeção utiliza tanto uma imagem de referência como o projeto da placa.

2.2.1 Inspeção de placa nua

Este processo é realizado após a impressão das trilhas sobre a placa virgem e antes da insersão dos componentes. O objetivo deste tipo de inspeção é garantir que a placa não possua problemas nas trilhas e *pads* e está adequada para a insersão dos componentes. Os tipos de defeitos mais procurados são (PUTERA; IBRAHIM, 2010) (LIN et al., 2009) (RAU et al., 2009) (LETA; FELICIANO; MARTINS, 2007) (MOGANTI et al., 1996):

- Furo faltando: esse tipo de problema ocorre somente em casos de produção de placas baseadas em THT. Trata-se da ausência de algum furo através do qual passaria o terminal de algum componente eletrônico;

- Circuito aberto: está associado ao rompimento ou falha na impressão de alguma trilha;

- Curto circuito: como o próprio nome indica, trata-se de um curto circuito entre alguma trilha ou *pad*.

2.2.2 Inspeção de placa montada

Este processo realiza-se após o processo de montagem, ou seja, após a insersão e soldagem dos componentes na placa e objetiva garantir que os componentes corretos tenham sido inseridos e estejam adequadamente posicionados e soldados. O processo de montagem encerra-se após a soldagem dos componentes, no entanto, a inspeção pode ser efetuada após a soldagem ou após a inserção. Os tipos de defeitos mais procurados após a etapa de inserção são (WU et al., 2010) (SUNDARAJ, 2009) (LEE; PARK, 2009) (WU; ZHANG; HONG, 2009) (LETA; FELICIANO, 2008) (OLIVEIRA; PIO, 2008) (CHO; PARK, 2008) (LIN; SU, 2006):

- Componente ausente: ausência de algum componente na placa;

- Componente rotacionado/invertido: componente inserido com polaridade invertida ou rotacionado em relação aos *pads*;

- Componente deslocado: componente deslocado em relação aos *pads*;

- Componente errado: insersão de algum componente errado (trocado) em lugar de outro.

É importante ressaltar que, apesar de os defeitos acima citados serem comumente procurados, os defeitos oriundos da soldagem são os que mais ocorrem nas linhas de produção, principalmente os defeitos de curto circuito e insuficiência de solda.

Existem ainda vários outros defeitos que não estão relacionados com o processo de montagem em si. São defeitos originados pelo inadequado manuseio, transporte ou armazenamento dos componentes ou placas, bem como, má qualidade dos mesmos. Esses fatores podem acarretar defeitos como placa empenada, componente com dimensões erradas, componente ou placa danificados, entre outros (DORO, 2004).

2.3 METROLOGIA ÓPTICA

O tipo de inspeção sem contato físico que tem sido largamente utilizado para inspeção de PCI é baseado em sistemas ópticos de inspeção (DORO, 2013). Esse tipo de inspeção é também caracterizado como sendo metrologia óptica (PFEIFER et al., 2003b). A metrologia óptica é considerada versátil, flexível e com alta precisão, possibilitando inspecionar pequenos detalhes em objetos. Além disso, como não há contato entre o dispositivo de medição e o objeto sendo medido, evita danos em ambos (PFEIFER et al., 2003b) (PFEIFER et al., 2003a). No caso de inspeção de PCI, o dispositivo óptico de medição que tem sido amplamente utilizado é a câmera, geralmente do tipo CCD (Charge-Coupled Device).

Considera-se importante o uso de metrologia óptica na inspeção de PCI visto que, tem sido extensivamente utilizada para essa finalidade, com bons resultados. Paralelo a isso, verifica-se que a metrologia óptica apresenta-se como uma tendência nas linhas de produção, visando garantia de qualidade (PFEIFER et al., 2003a).

2.4 TRABALHOS RELACIONADOS

A seguir são apresentados trabalhos relacionados à inspeção de PCI, juntamente com uma análise a respeito das limitações observadas. Todos os trabalhos apresentados baseiam-se na utilização de câmeras e no desenvolvimento de algum software de PDI. Na literatura relativa ao tema, encontra-se vários trabalhos que se referem a "inspeção visual". Essa expressão significa que o sistema de inspeção proposto está baseado no uso de visão computacional.

Visão computacional, também conhecida como visão de máquina, trata da extração de informações úteis a partir de imagens. Nesse contexto, imagens podem ser interpretadas como projeções bidimensionais do mundo real, que é tridimensional (JAIN; KASTURI; SCHUNCK, 1995). Com o desenvolvimento da área de processamento de imagens, muitas novas técnicas tem sido agregadas à mesma. Com isso, várias técnicas que poderiam ser atribuídas à visão computacional, tem sido atribuídas à área de PDI. Entretanto, a área de visão computacional faz uso de técnicas de várias áreas como, por exemplo, inteligência artificial e percebe-se que há sempre o envolvimento de algum aspecto cognitivo (JAIN; KASTURI; SCHUNCK, 1995).

O trabalho de (WU; ZHANG; HONG, 2009) e (WU et al., 2010) foca na inspeção de placas SMD montadas, mais especificamente, no posicionamento de componentes que apresentam dois eletrodos, como resistores e capacitores. Os defeitos são divididos em quatro classes: componente ausente, componente rotacionado, componente deslocado e componente errado. Os defeitos são detectados a partir de informações sobre a posição dos eletrodos e sobre as características da cor de cada componente. Para reduzir o tempo computacional existe uma etapa de treinamento, na qual é utilizado um classificador bayesiano para identificar as classes dos componentes, baseando-se na cor (RGB) destes. Essa abordagem é adequada para inspecionar a montagem de PCI que utilizam componentes sem serigrafia. Portanto, componentes do tipo CI (Circuito Integrado) não estão cobertos por esse tipo de técnica.

O sistema de inspeção desenvolvido por (SUNDARAJ, 2009) está baseado no uso de subtração de imagens, visando encontrar defeitos como ausência de componentes e componentes desalinhados. A ideia principal é realizar uma fase de treinamento, na qual o *background* será "aprendido". Após isso, procura-se pelos *pixels* que não pertencem ao *background*. Inicialmente há uma etapa de treinamento, efetuado utilizando-se uma amostragem de 100 *frames* (imagens) adquiridos a partir de uma PCI perfeita. O autor considera que esses 100 *frames*

são suficientes para representar as variações que podem ocorrer no *background*. Essa etapa de treinamento é realizada basicamente a partir de somas e produtos vetoriais. Os vetores de entrada para essa etapa são os *pixels* da imagem, ou seja, cada *pixel* é um vetor contendo três elementos, que são os respectivos valores dos canais RGB que representam a cor do *pixel* em questão. Para efetuar a discriminação entre um *background-pixel* e um *não-background-pixel*, é utilizada uma função gaussiana para calcular a probabilidade de um *pixel* pertencer ao *background*. Essa função gaussiana utiliza parâmetros que foram estimados a partir dos vetores calculados na etapa inicial de treinamento. Após computar essa função gaussiana, é utilizado um *threshold*, empiricamente definido. Se o resultado do cômputo dessa função estiver acima desse *threshold*, o *pixel* é considerado pertencente ao *background*. Segundo o autor, esse *treshold* necessita ser calculado empiricamente pela dependência de fatores como câmera, sistema de iluminação, *setup* da montagem e tipo de PCI sendo inspecionada. Caso seja detectado um alto percentual de *pixels* que não pertençam ao *background*, será considerado que, naquela região da imagem, há um componente ausente ou desalinhado. Uma limitação dessa abordagem é que não há uma forma de diferenciar se um componente está desalinhado ou ausente. Outro problema é que, a presença de alguma marcação de fábrica em algum componente, diferente do que havia na imagem utilizada como treinamento, será interpretado como defeito. Outra restrição desse método é quanto às imprecisões no posicionamento da PCI sendo inspecionada, bem como, dos componentes montados sobre a mesma. Esses fatores poderão acarretar a identificação de falsos defeitos. O autor sugere que, pare contornar esses problemas a nível industrial, sejam utilizados laseres posicionadores.

O trabalho de (OLIVEIRA; PIO, 2008) objetiva detectar a ausência de componentes em PCI. Inicialmente é utilizada uma máscara binária em que *pixels* com valor "1" representam os *pads* da região da imagem onde seria soldado um componente e *pixels* "0" representam a região externa aos *pads*. Após aplicar-se uma operação lógica *AND* entre essa máscara binária e a imagem de uma placa perfeita, também binarizada, é realizado o treinamento de uma rede bayesiana. Para testar a presença de componentes, verifica-se a quantidade de *pixels* com valor "1" resultantes da mesma operação *AND*, porém, agora realizada entre a máscara binária e a imagem de uma placa de teste. Com o uso de redes bayesianas, a detecção de defeitos é baseada no cálculo da probabilidade de uma determinada região da imagem pertencer à hipótese de componente presente, calculada previamente a partir da imagem da placa

perfeita. A utilização de imagens binarizadas faz com que a precisão do método apresentado seja afetada por variações de iluminação, causando a "migração" de *pixels* entre as regiões separadas pelo *threshold*. Outro problema ainda acarretado pela binarização é que, existem componentes de cores variadas e ocasionalmente pode haver um componente com uma cor que acabe sendo alocada pelo *threshold* como sendo o fundo da PCI, impossibilitando a detecção de defeitos. Com isso, seria importante o uso de vários *thresholds*, o que poderia dificultar o *setup* do sistema para PCI com muitos componentes diferentes.

O sistema proposto por (LETA; FELICIANO, 2008) avalia tanto placas sem componentes como placas montadas, porém, em etapas distintas. Nas placas sem componentes, é utilizado o conceito de conectividade de pontos (furos) e também é verificada a presença de curtos ou quebras nas trilhas, entre outros defeitos. Na análise das placas montadas, é utilizado o cálculo de correlação de imagens, juntamente com a informação prévia sobre a posição de cada componente. Somente utilizando correlação de imagens não é possível identificar o tipo de defeito encontrado. Com isso, os autores propõem o uso de outras técnicas como, por exemplo, cálculos utilizando o histograma da imagem. Entretanto, os autores não detalham sobre o uso dessas técnicas. Duas restrições que pode-se destacar sobre o uso de correlação de imagens neste caso é que, variações na iluminação, mesmo que lineares, e imperfeições na superfície dos componentes, afetam o coeficiente de correlação, reduzindo a precisão do sistema e induzindo a erros. Uma outra limitação desta abordagem é que há a necessidade de se obter previamente o coeficiente de correlação e como esse valor pode variar entre as placas, para cada modelo de placa a ser inspecionada, esse valor necessita ser recalculado.

Utilizando um foco diferente, o trabalho de (CHO; PARK, 2008) apresenta um método de inspeção, cujo objetivo principal é a redução do volume de dados a ser processado. Com o uso de transformada wavelet, as imagens são transformadas em estruturas com cerca de 7% do volume de informação das imagens originais. Isso faz com que seja necessário menos espaço em disco para armazenar as imagens, menos memória RAM utilizada, bem como, menos processamento. A detecção de defeitos é baseada no uso de *template matching* entre uma foto da região do componente a ser inspecionado, obtida a partir uma placa perfeita, e a imagem da mesma região, obtida a partir de uma placa de teste. A técnica de *template matching* utilizada é baseada em correlação de imagens, similar a *mean absolute difference* (MAD) e *mean square difference* (MSE). Porém, antes do processo de *matching* é efetuado o

seguinte processamento nas imagens, em sequência: aplicação da transformada wavelet discreta, binarização e segmentação. Os defeitos que são procurados com esse método são: componente ausente, componente errado, componente desalinhado e componente levantado (tombstone), no entanto, os autores não detalham sobre a precisão dos resultados obtidos com o método. Nota-se ainda que o processo de binarização exige a definição prévia de um *threshold*. Apesar de não detalhado pelos autores, essa etapa é importante no processo de inspeção e como tal, implica em aumentar o tempo de *setup* do sistema para se encontrar um *threshold* ideal para a PCI em questão.

Baseados na comparação entre assinaturas de imagens (*hashcode*), os autores de (LONGJIANG; YING; SHENGHE, 2007) apresentam um método para inspeção de PCI que visa encontrar os seguintes defeitos: componente ausente, componente errado e componente invertido. Tendo calculado o *hashcode* da imagem de um componente de referência (a partir de uma placa perfeita), o *hashcode* desse componente é comparado com o *hashcode* de cada imagem de teste. Como medida de similariade, é utilizada uma versão modificada da distância de Hausdorff, entretanto, os autores não comentam a respeito das modificações efetuadas. Os defeitos são detectados e identificados com base no valor da distância de Hausdorff. Para cada possível defeito é associada uma faixa de possíveis valores para a distância. Uma limitação dessa abordagem é que, dependendo do quão grave for um certo defeito, o valor da distância de Hausdorff pode se aproximar da faixa de distância de um tipo diferente de defeito. Conforme apresentado pelos autores nos experimentos realizados pelos mesmos, o valor da distância de Hausdorff para um defeito de translação em 12x15 *pixels* foi de "0,1264" e para um defeito de rotação em 5 graus foi de "0,1222". Observando que o valor da distância de Hausdorff para um defeito de rotação em 90 graus foi de "0,1998", percebe-se que o valor encontrado para o defeito de translação está muito próximo ao valor encontrado para o defeito de rotação em 5 graus. Outra limitação desse método é que a presença de variações na iluminação, bem como, ranhuras na superfície dos componentes, podem alterar significativamente o *hashcode*, induzindo a erros na inspeção, principalmente relativos ao tipo de defeito encontrado.

Visando detectar os defeitos de componente deslocado e componente ausente em inspeção de PCI, o trabalho de (OPREA et al., 2007) inspira-se na ideia da percepção de movimento em uma sequência de imagens. A percepção de movimento é simplificada para a simples detecção de mudanças no valor de um determinado *pixel*, em nível de cinza, entre dois *frames*. Para ressaltar o "movimento" entre *frames*,

uma imagem RGB é construída de forma que cada canal corresponde ao valor do canal de brilho da mesma imagem no formato HSV. Sendo assim, o valor do brilho (canal V) de um *pixel* da foto de uma placa perfeita é usado para formar o canal R e o valor do brilho da foto de uma placa de teste é usado para formar os canais G e B. Após isso, a imagem é novamente convertida pra o formato HSV e o canal S é utilizado para identificar os defeitos. Para uma localização mais precisa do defeito, é realizado um *threshold* nesse canal. Igualmente aos demais trabalhos relatados anteriormente, o estabelecimento do valor de *threshold* pode ser trabalhoso. Outra observação relevante é que este método é muito sensível a ruídos nas imagens ocasionados por, por exemplo, variações na iluminação.

Em (LIN; SU, 2006) é proposto um sistema de inspeção com duas etapas: uma etapa de filtro baseada em correlação de imagens e uma etapa de classificação baseada em redes neurais. A etapa de filtro objetiva diminuir o número de imagens a serem classificadas na segunda etapa, consequentemente reduzindo o tempo de processamento. Na segunda etapa são utilizados diferentes tipos de técnicas simples como, por exemplo, análise de histograma. Para cada técnica utilizada, é gerado um índice, que serve como entrada para a rede neural. Os autores sugerem a utilização de cinco índices, visando reduzir o percentual de falsos negativos. Como resultado, o sistema proposto gera na saída da rede neural a classificação da imagem em, componente correto, ausente, deslocado ou invertido. Uma deficiência da abordagem proposta é que, para cada tipo de componente, é necessário estabelecer-se manualmente um *threshold*, para ser utilizado na primeira etapa, tornando esse processo oneroso. Dentre as várias técnicas testadas, os autores sugerem a utilização de correlação de imagens na primeira etapa, objetivando reduzir a quantidade de falsos negativos. Entretanto, essa técnica é sensível à variações na iluminação, bem como, ranhuras na superfície dos componentes. Dessa forma, observa-se que, para evitar uma grande quantidade de falsos negativos na primeira etapa, é necessário que o coeficiente de correlação utilizado como *threshold* seja bem amplo (permissivo), reduzindo a eficácia da primeira etapa como filtro. Outra ressalva que se faz é que a classificação baseada em redes neurais é altamente dependente do conjunto de imagens utilizado no treinamento. Com isso, ressalta-se que para se obter bons resultados, é necessário utilizar um grande conjunto de imagens de componentes do mesmo tipo.

Muitos dos trabalhos relatados utilizam um procedimento de binarização. Esse procedimento requer algum critério de *threshold* para

separar os *pixels* em escala de cinza. Conforme apresentado em (DEMIR et al., 1994), a maioria dos métodos de *threshold* não apresenta bons resultados para o reconhecimento de caracteres. Portanto, a binarização deve ser utilizada com muita cautela em inspeção da serigrafia de componentes eletrônicos, que esteja baseada em reconhecimento de caracteres.

Nos últimos anos, observa-se uma tendência de pesquisa em inspeção visual de PCI, focando na inspeção de placa montadas. Isso se deve ao fato de que, há muito tempo se pesquisa a respeito da inspeção de placas sem componentes e muito progresso já foi realizado nessa área (MOGANTI et al., 1996). No entanto, novas técnicas tem sido propostas, sejam relacionadas à inspeção em si (PUTERA; IBRAHIM, 2010) (LIN et al., 2009) (RAU et al., 2009) (IBRAHIM; AL-ATTAS; ASPAR, 2002) (KIM; PYUN, 2001), sejam relacionadas a outros aspectos como, por exemplo, melhoramento da qualidade de imagens (GUAN; GUO, 2011) (IBRAHIM et al., 2008). Como o foco deste trabalho está voltado à inspeção de placas montadas, as técnicas de inspeção de placas antes da montagem dos componentes não são detalhadas. Para essa finalidade, sugere-se a leitura do trabalho de (MOGANTI et al., 1996), que apesar de relativamente antigo, apresenta um amplo resumo das técnicas desenvolvidas com esse enfoque, incluindo a análise de algumas soluções comerciais. Pode-se ainda citar outros trabalhos com um enfoque ainda diferente: são as técnicas destinadas especificamente à inspeção da solda em placas montadas (LOH; LU, 1999).

2.5 TEMPLATE MATCHING

Diversas abordagens para inspeção de PCI, algumas apresentadas no item 2.4, utilizam alguma forma de *template matching*, que é uma técnica de grande importância na área de visão computacional e pode ser considerada um gargalo nessa área (WU; DAI, 2005). Muitas vezes há na literatura uma diferenciação entre *image matching* e *template matching*. *Image matching* é um método de comparação entre duas imagens na tentativa de "casar" elementos entre ambas. Trata-se de um procedimento que utiliza duas imagens, diga-se A e B, e objetiva encontrar em B, um ou mais elementos que estão presentes na imagem A. Esses elementos podem ser objetos ou até mesmo uma cena (*matching* usando uma imagem inteira). No caso do *template matching*, o mesmo pode ser interpretado como um tipo de *image matching* em que uma das imagens, por exemplo, a imagem A contém um template para

alguma características que se deseja encontrar em B. O template pode ser, por exemplo, a foto ou o contorno de algum objeto. É bastante comum encontrar-se trabalhos relacionados a *template matching* em que o procedimento de *matching* utiliza apenas imagens como dados de entrada. Por isso, em muitos casos o *template matching* é apresentado com a mesma semântica de *image matching* (WU; DAI, 2005) (JURIE; DHOME, 2002) (BRUNELLI, 2009). O aspecto mais importante de *template matching* e *image matching* é o procedimento do *matching* em si. Esse procedimento trata-se da verificação de similaridade ou diferença entre elementos (BRUNELLI, 2009). Em geral, após um processo de *matching*, tem-se a associação de um elemento da imagem A a um elemento da imagem B, baseado na similaridade entre ambos. Entretanto, também é muito comum encontrar sistemas que buscam várias instâncias de um mesmo elemento da imagem A na imagem B, ou seja, realiza-se um *matching* de forma que o resultado pode ser uma associação 1x1 ou 1xN.

De uma maneira geral, pode-se dividir as técnicas de *template matching* em dois grupos, em função das informações utilizadas:

- Baseados em regiões: o processamento é feito diretamente nos *pixels* das imagens, ou em pequenas variações destes. Os algoritmos SAD (Sum of Absolute Differences) e SSD (Sum of Squared Differences) são exemplos que se baseiam diretamente no valor dos *pixels* (SHIN; MOON, 2009) (JAIN; KASTURI; SCHUNCK, 1995):

$$SAD = \sum_{[i,j] \in R} |f - g| \tag{2.1}$$

$$SSD = \sum_{[i,j] \in R} (f - g)^2 \tag{2.2}$$

  Sendo que, g[i,j] é um template, f[i,j] é uma imagem de teste e R é a região do template. Quanto menores os valores de SAD e SSD, mais semelhantes serão as imagens, na região analisada.

- Baseados em características: o processamento é realizado sobre dados obtidos a partir dos *pixels* (após um pré-processamento) e não diretamente sobre os mesmos. Esse processo é normalmente mais robusto, visto que não está tão sujeito às variações nos valores dos *pixels* (normalmente em tons de cinza). Algoritmos amplamente difundidos como o SURF (Speeded Up Robust Features) e o SIFT (Scale Invariant Feature Transform) são exemplos

de algoritmos baseados em características, que geram estruturas denominadas "keypoints", estas utilizadas nos *matchings* (BAY et al., 2008) (LOWE, 2004). As estruturas utilizadas em *matchings* baseados em características podem ser mais simples do que os *keypoints* gerados pelo SURF e pelo SIFT. Segmentos de linha e bordas são exemplos de estruturas que também pode ser utilizadas (JURIE; DHOME, 2002).

2.6 ALGORITMO SIFT

O algoritmo SIFT (LOWE, 2004), desenvolvido por David G. Lowe, é uma das mais relevantes técnicas de *image matching* atuais. Isso evidencia-se pela elevada quantidade de trabalhos acadêmicos que têm sido realizados com a utilização do SIFT, bem como, pela distribuição de uma implementação desse algoritmo junto ao pacote de bibliotecas de processamento de imagens OpenCV (TEAM, 2013). Não obstante, a combinação da alta distintividade das *SIFT-features*, juntamente com a invariância à escala e o bom desempenho, tornam o SIFT adequado para aplicações reais. Em resumo, o objetivo do SIFT é, a partir de imagens, gerar features denominadas *keypoints*, com as seguintes características:

- Altamente distintivas;
- Invariantes à escala;
- Invariantes à rotação;
- Parcialmente invariantes à iluminação;
- Parcialmente invariantes à variações 3D no ângulo de visão da câmera de captura das imagens.

A distintividade torna as *SIFT-features* úteis no reconhecimento de objetos ou cenas pois, as *features* podem ser "casadas" corretamente, com uma considerável probabilidade, em relação a um grande banco de dados de *features*. A figura 4 apresenta a foto de um ambiente de escritório, com diversos círculos representando a localização das *SIFT-features* geradas a partir dessa foto.

O SIFT em sua versão atual (2004), é o resultado do aprimoramento de versões anteriores do mesmo algoritmo, elaboradas pelo mesmo autor (LOWE, 1999)(LOWE, 2001)(BROWN; LOWE, 2002). Após

Figura 4 – Foto de um ambiente de escritório, com círculos representando a localização dos *keypoints* gerados.

a publicação da última versão do algoritmo (2004) é que o mesmo passou a ser amplamente utilizado e estudado.

A extração das *SIFT-features* não é realizada em um único passo. Após a detecção inicial de *keypoints*, são realizados vários processamentos visando tornar os *keypoints* mais robustos a ruídos e manter somente aqueles que atendam a uma série de critérios. Para isso, o algoritmo é dividido em etapas e as principais são:

1. Detecção de extremos no espaço de escala;
2. Localização e eliminação de *keypoints*;
3. Atribuição de orientação aos *keypoints*;
4. Geração dos descritores.

Apesar dessas etapas serem divididas em sub-etapas e em cada qual sendo executadas diversas operações, o algoritmo apresenta um

bom desempenho. Isso ocorre porque as operações são realizadas sequencialmente (em cascata), com as operações mais pesadas sendo realizadas após um filtro inicial.

De acordo com Lowe, o SIFT gera uma grande quantidade de *keypoints*, que abrangem amplamente uma imagem a nível de localização e escala. Apesar de isso ser verificado na prática, é um dado bastante relativo pois, a quantidade de *keypoints* depende da aparência da imagem. Em muitas imagens reais, devido à aparência relativamente constante (sem grandes variações de contraste) de alguns objetos, poucos *keypoints* são gerados. Para contornar esse problema, os parâmetros do SIFT podem ser variados de forma que, para certos domínios, os filtros sejam relaxados, acarretando a geração de um maior número de *keypoints*. Isso deve ser considerado, pois a quantidade de *keypoints* é um fator importante no reconhecimento de objetos, visto que Lowe sugere que um reconhecimento confiável só é possível com 3 ou mais *keypoints* corretamente "casados".

Sabendo-se que os *keypoints* são altamente distintivos entre si e tendo-se uma elevada quantidade de *keypoints* em uma base de dados, é necessário um método para encontrar *keypoints* específicos nessa base de dados. Mais adiante são abordadas as técnicas propostas para procurar *keypoints* em uma base de dados. Entretanto, Lowe sugere a distância Euclidiana como medida de similaridade entre dois *keypoints*, o que parece natural, visto que o descritor de um *keypoint* é um vetor de elementos.

2.6.1 Detecção de extremos no espaço de escala

O primeiro estágio da detecção de *keypoints* trata-se da identificação de locais e escalas que possuam boa repetibilidade, mesmo na ocorrência de variações no ângulo de visão de um mesmo objeto. Nesse estágio busca-se locais distintivos, de forma que possam ser detectados mesmo após variações de escala.

Uma forma eficiente encontrada por Lowe para identificar *features* invariantes à escala (locais invariantes à escala) é usar um *kernel* gaussiano, conforme mostrado na equação 2.3.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.3}$$

Uma operação de convolução ($*$) é aplicada à imagem inicial I utilizando G. Essa operação é aplicada para cada nível diferente de σ

e gera uma imagem L, conforme expresso na equação 2.4.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.4}$$

Para se obter os resultados desejados na detecção de máximos e mínimos, seria necessário utilizar o operador laplaciano da gaussiana, porém, essa é uma operação muito custosa. Sendo assim, para reduzir o custo computacional, é utilizada uma simples operação de subtração de imagens L, entre duas imagens em escalas adjacentes, dentro de uma mesma oitava. Esse processo é ilustrado na figura 5.

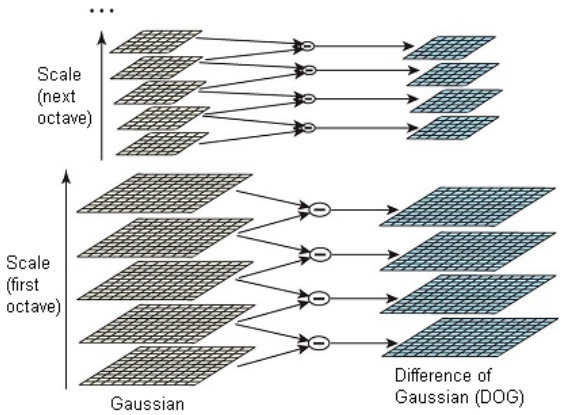


Figura 5 – Subtração de gaussianas nas diferentes oitavas do espaço de escala. Fonte: (LOWE, 2004).

A coluna "Gaussian" na figura 5 representa as imagens L. Cada imagem L de uma mesma oitava possui um diferente nível de suavização (σ), proporcional à respectiva oitava. Todas as imagens L de uma mesma oitava possuem o mesmo tamanho, que corresponde à metade do tamanho das imagens da oitava anterior. Esse processo está expresso na equação 2.5:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{2.5}$$

Cada imagem em D representa a subtração de duas imagens L adjacentes, separadas por um fator de escala k. A coluna "Difference of

Gaussian" na figura 5 ilustra a formação das imagens D. É importante ressaltar que essa operação de subtração de gaussianas é uma aproximação do laplaciano da gaussiana normalizado pela escala, $\sigma^2\nabla^2 G$, cujos máximos e mínimos correspondem às mais estáveis features, se comparado com outras técnicas como, por exemplo, "Harris corner detector" (LOWE, 2004).

Uma vez gerado o conjunto de imagens através da subtração de gaussianas (DoG), essas imagens passam a ser utilizadas nas próximas etapas. A identificação de quais *pixels* da imagem serão candidatos a *keypoints* inicia com a detecção de máximos e mínimos locais nas imagens DoG. Para isso, cada *pixel* sendo analisado é comparado a seus vizinhos, considerando uma região de 3x3 *pixels*. Dessa forma, um determinado *pixel* sendo analisado é comparado aos 8 *pixels* vizinhos na imagem em questão e a 9 vizinhos nas escalas adjacentes acima e abaixo. Esse processo é ilustrado na figura 6.

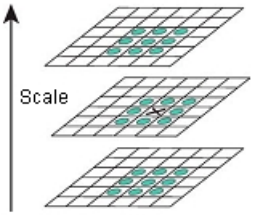


Figura 6 – Detecção de máximos e mínimos na vizinhança de 8 de um determinado *pixel*, marcado com X. Fonte: (LOWE, 2004).

Um determinado *pixel* será candidato a *keypoint* caso seja maior ou menor do que todos os seus 26 vizinhos verificados nesse processo.

2.6.2 Localização e eliminação de keypoints

Para tornar mais precisa a localização dos *keypoints* e eliminar aqueles que possuem baixo contraste, é realizada uma aproximação a *sub-pixel*, utilizando expansão por série de Taylor até o termo de segunda ordem, conforme a equação 2.6.

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{2.6}$$

onde $\mathbf{x} = (x, y, \sigma)^T$ é o deslocamento em relação ao ponto de amostra (*keypoint*). A expansão é realizada considerando o ponto de amostra como sendo a origem, dessa forma o cálculo é feito similarmente à uma série de Maclaurin, em que o ponto a partir do qual a série é expandida, é igual a 0 ($x_0 = 0$, $y_0 = 0$, $\sigma_0 = 0$).

A localização de um extremo, máximo ou mínimo, é obtida derivando-se $D(\mathbf{x})$ em relação a $\mathbf{x}$ e igualando-se a 0, conforme apresentado na equação 2.7:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}} \tag{2.7}$$

Caso o valor de $\hat{\mathbf{x}}$ seja maior do que 0,5 em qualquer dimensão, será considerado que o extremo está mais próximo a outro *keypoint* e o procedimento de refinamento da localização será cancelado para o *keypoint* em questão.

Uma vez que o *offset* $\hat{\mathbf{x}}$ foi calculado em relação à origem, para determinar a posição final do *keypoint*, esse valor será adicionado à localização inicial do mesmo, ou seja, a (x_0,y_0,σ_0), que inicialmente fora definido como 0 nas 3 dimensões.

O valor final (em escala de cinza) de um *keypoint* é obtido através o uso da função $D(\mathbf{x}) = (x, y, \sigma)$, considerando a nova localização do *keypoint*, $\hat{\mathbf{x}}$. Isso é feito, substituindo-se a equação 2.7 em 2.6:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2} \frac{\partial D^T}{\partial \mathbf{x}} \hat{\mathbf{x}} \tag{2.8}$$

O valor de $D(\hat{\mathbf{x}})$ obtido com a equação 2.8 é utilizado para eliminar *keypoints* com baixo contraste. Nos experimentos realizados por Lowe, foi utilizado o *threshold* de 0,03 para $D(\hat{\mathbf{x}})$ (considerando valores no intervalo [0,1]). Entretanto, o valor adequado para esse *threshold* pode variar em função do conjunto de imagens utilizado e é um dos parâmetros do SIFT.

Em resumo, a aproximação para *sub-pixel* trata-se de um reposicionamento (refinamento) dos *keypoints* em localizações entre *pixels*, ou seja, localizações não-inteiras. Dessa forma, ao invés de os *keypoints* estarem localizados em posições exatas de *pixels* como, por exemplo, (20;37) ou (15;60), localizam-se em posições tais como, (20,1;37,2). De acordo com Lowe, isso torna os *matchings* mais estáveis.

Mesmo após a eliminação de *keypoints* com baixo contraste, alguns *keypoints* ainda precisam ser eliminados, visando estabilidade. A subtração de gaussianas resulta em valores altos para *keypoints* localizados em bordas, inclusive para aqueles em uma localização ruim e

sensível a ruídos. Baseando-se no detector de cantos de Harris (Harris corner detector), verifica-se a proporção entre as principais curvaturas. Isso é realizado utilizando-se uma matriz hessiana de 2x2, **H**, calculada na localização e escala do *keypoint*, conforme abaixo:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.9}$$

Considerando que os autovalores são proporcionais às principais curvaturas e que necessita-se apenas da proporção entre elas para eliminar os *keypoints*, os autovalores não precisam ser explicitamente calculados. Para então eliminar os *keypoints* com má localização, verifica-se a razão entre as principais curvaturas, conforme a inequação 2.10

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r} \tag{2.10}$$

em que, $Tr(\mathbf{H})$ é o traço da matriz **H** e $Det(\mathbf{H})$ é o determinante da mesma. O parâmetro r corresponde à proporção entre o maior e o menor autovalor da matriz **H**. Lowe utilizou $r = 10$ em seus experimentos, o que significa eliminar *keypoints* cuja razão entre as principais curvaturas seja maior do que 10.

Conforme apresentado por Lowe, a resposta da principal curvatura será alta quando computada com *pixels* que atravessem uma borda e será baixa quando os *pixels* estiverem na direção da borda. Essa informação é relevante pois, as derivadas da matriz **H** são computadas através da subtração de *pixels* próximos ao *keypoint*.

2.6.3 Atribuição de orientação aos keypoints

Visando atingir invariância à rotação na imagem, é atribuída uma orientação a cada *keypoint*. Para isso, é avaliada a região ao redor do *keypoint*, utilizando as imagens D, geradas após o processo de convolução com o *kernel* gaussiano, conforme a equação 2.5.

Para cada *pixel* ao redor do *keypoint*, são calculadas suas respectivas magnitude $m(x, y)$ e orientação $\theta(x, y)$, conforme as equações 2.11 e 2.12, respectivamente.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{2.11}$$

$$\theta(x,y) = tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)} \right) \quad (2.12)$$

Utilizando a orientação do gradiente ao redor do *keypoint*, é criado um histograma de orientação a cada 10 graus, ou seja, contendo 36 possíveis orientações. Cada magnitude e orientação calculada ao redor do *keypoint* é adicionada ao histograma. A orientação determina a posição em que será adicionada no histograma e a magnitude, o valor a ser adicionado. A figura 7 ilustra a formação desse histograma. Entretanto, antes de ser adicionada ao histograma, a magnitude é ponderada utilizando-se uma janela circular gaussiana, com σ 1,5 vez o valor do σ na escala do *keypoint*.

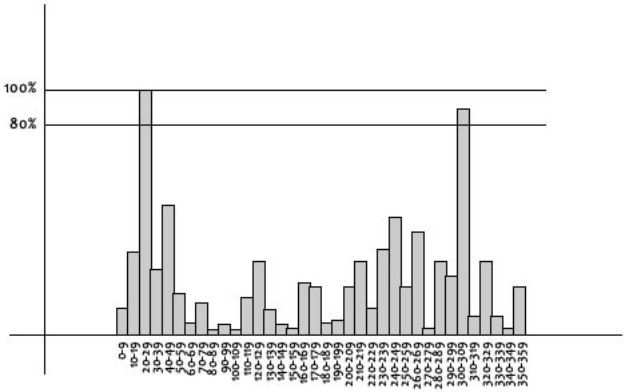


Figura 7 – Representação de um possível histograma utilizado para determinar a orientação de um *keypoint*. Fonte: (SINHA, 2013).

O maior pico do histograma determinará a orientação do *keypoint* e qualquer outro pico cujo valor seja maior ou igual a 80% do valor do maior pico, será utilizado para criar outro *keypoint* com as mesmas posição e escala, porém, com a orientação determinada por esse outro pico. No exemplo da figura 7, a orientação do *keypoint* estará entre 20 e 29 graus, porém, como surgiu um outro pico no histograma com magnitude acumulada superior a 80%, será gerado um outro *keypoint*, com orientação entre 300 e 309 graus.

2.6.4 Geração dos descritores

A última etapa do SIFT trata da geração do descritor do *keypoint*. Para gerar o descritor, novamente são utilizados os gradientes da região ao redor do *keypoint*. No entanto, nesta etapa a obtenção da magnitude e da orientação dos gradientes são realizadas relativas à orientação do *keypoint*, calculada anteriormente.

De forma semelhante à utilizada para obter a orientação do *keypoint*, é aplicada uma janela circular gaussiana pra atribuir um peso à magnitude de cada gradiente. No entanto, essa segunda suavização gaussiana utiliza σ igual à metade do tamanho da largura da janela de composição do descritor. A figura 8 ilustra a janela de 16x16 *pixels* obtidos como amostra ao redor do *keypoint*. À esquerda dessa figura é possível observar uma representação da janela gaussiana utilizada para ponderar a magnitude de cada amostra. À direita dessa figura observa-se a estrutura utilizada para compor o descritor, sendo que, a direção de cada seta indica a orientação do respectivo gradiente e o comprimento indica a sua magnitude.

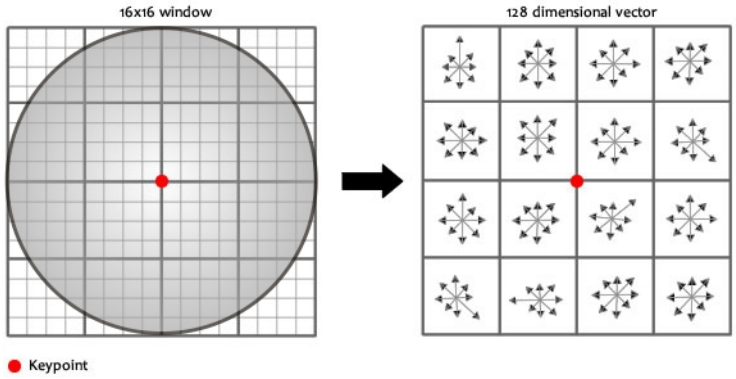


Figura 8 – Representação da janela de amostras computada ao redor do *keypoint* para formar o descritor. Fonte: Adaptado de (SINHA, 2013).

Fazendo uso da direção e magnitude de cada gradiente, são criados 16 histogramas de orientação, com 8 orientações em cada, conforme ilustrado à direita da figura 8. A magnitude de cada gradiente é somada à sua orientação no histograma, similarmente ao realizado para

obter a orientação do *keypoint*.

Utilizando esses 16 histogramas, é criado o descritor, que é um vetor de 128 elementos (4x4x8). Inicialmente, cada elemento do descritor corresponde ao valor de uma posição na matriz de histogramas.

Após a geração do descritor, o mesmo é normalizado para um vetor de comprimento unitário. Essa operação cancela os efeitos de possíveis alterações na imagem, que correspondam a multiplicar cada elemento do vetor por uma constante. Visando reduzir variações não lineares na iluminação, cada elemento do descritor é limitado a um valor máximo de 0,2 e o vetor é novamente normalizado para um comprimento unitário. Esse valor de 0,2 foi obtido experimentalmente pelo autor do SIFT.

Com a aplicação desses procedimentos, o descritor torna-se invariante a variações afim na iluminação e parcialmente invariante à variações não-lineares na mesma.

Após a aplicação do SIFT à uma imagem tem-se um conjunto de *keypoints*. A quantidade de *keypoints* extraídos de uma imagem depende do conteúdo da mesma, bem como, dos valores atribuídos aos parâmetros do algoritmo. Entretanto, de acordo com o autor, uma típica imagem 500x500 *pixels* irá gerar aproximadamente 2.000 *keypoints*.

Uma das aplicações mais comuns para features extraídas de imagens é o reconhecimento de objetos. Para essa finalidade, Lowe sugere as seguintes etapas, após a geração de uma base de dados de *keypoints*:

- Realizar um *matching* para todos os *keypoints* obtidos a partir de uma imagem de teste. Como medida de similaridade para um *match*, é utilizada a distância euclidiana;

- Eliminar os *matches* cuja razão entre a maior e a segunda maior distância euclidiana seja superior a 0,8. Quanto menor for essa razão, um *match* será mais seguramente um bom *match*, porém, mais bons *matches* serão descartados. De acordo com o autor, para essa razão de 0,8, menos de 5% de bons *matches* serão descartados, enquanto que, cerca de 90% de falsos *matches* serão eliminados. Para buscar o *keypoint* mais próximo e o segundo mais próximo na base de dados, é utilizado o algoritmo Best-Bin-First (BBF) (BEIS; LOWE, 1997), que é uma adaptação do algoritmo K-D Tree (FRIEDMAN; BENTLEY; FINKEL, 1977). BBF retorna o elemento mais próximo com alta probabilidade e permite tornar eficiente a busca, ao possibilitar que uma busca seja encerrada após um certo número de elementos verificados;

- Realizar clusterização dos *matches* usando a transformada generalizada de Hough (BALLARD, 1981). Segundo Lowe, a clusterização é importante para que se possa descartar *matches* entre *keypoints* que pertençam a objetos diferentes, mas que passaram no teste anterior (baseado na distância euclidiana). Utilizando os parâmetros de localização (x,y), escala e orientação do *keypoint* obtido (encontrado) em cada *match*, realiza-se uma votação para tentar descobrir a localização, escala e orientação do objeto. Todos os *matches* que votaram em *clusters* com menos de 3 votos devem ser descartados, conforme sugerido por Lowe;

- Utilizando os *clusters* com pelo menos 3 votos, os *keypoints* são submetidos a uma transformação geométrica entre as imagens envolvidas no *matching*. Para obter os parâmetros ótimos dessa relação, é utilizado o método de mínimos quadrados a partir da equação 2.13. Nessa equação, "x,y" representa a localização de um *keypoint* na imagem de referência, a partir da qual foi gerado o banco de dados de *keypoints*. A localização do *keypoint* na imagem de teste (ponto transformado) é representado por "u, v", Os parâmetros da transformação afim, em termos de escala, rotação e deformação (stretch), são representados por "$m_1 ... m_4$". A translação nos eixos X e Y é representada por "t_x, t_y", respectivamente.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{2.13}$$

  Escrevendo a equação 2.13 na forma "$\mathbf{Ax} = \mathbf{b}$" e resolvendo a equação normal 2.14, obtem-se os parâmetros ótimos da transformação. Na equação 2.14, todos os parâmetros da transformação estão reunidos na matriz "$\mathbf{x}$".

$$\mathbf{x} = \left[\mathbf{A^T A}\right]^{-1} \mathbf{A^T b} \tag{2.14}$$

  Matches cujo erro de posicionamento for maior do que a metade do *range* usado na divisão dos *bins* da transformada de Hough, serão eliminados. Novamente, os *clusters* restantes com menos de 3 *matches* (votos) serão eliminados. Após isso, a solução por mínimos quadrados é aplicada novamente;

- Por último, é realizada uma verificação da probabilidade de ocorrência de falsos *matches*. Com essa probabilidade, realiza-se uma

análise bayesiana, que determinará a probabilidade do objeto estar presente na imagem (LOWE, 2001). Lowe sugere uma probabilidade de 98% para um reconhecimento confiável. A probabilidade de ocorrência de falsos *matches* depende do tamanho do objeto (imagem do objeto), bem como, da densidade de *keypoints* na região da imagem do objeto.

2.6.5 Variações do SIFT

Além de ser amplamente utilizado, algumas melhorias foram propostas ao SIFT. Em (KE; SUKTHANKAR, 2004) é apresentado o PCA-SIFT, uma nova forma de computar o descritor do *keypoint*. No PCA-SIFT, a única modificação feita em relação ao SIFT padrão é a redução da dimensão do descritor, através da utilização de PCA (Principal Components Analysis - Análise de Componentes Principais) para a geração do mesmo. Nos experimentos apresentados em (KE; SUKTHANKAR, 2004), é possível reduzir a dimensão do descritor para 20 e ainda realizar *matchings* mais precisos que os realizados com o descritor padrão do SIFT, que possui dimensão de 128. Para uma comparação mais detalhada entre os algoritmos, sugere-se a leitura do artigo de Mikolajczyk e Schmid (MIKOLAJCZYK; SCHMID, 2005), no qual é apresentada uma comparação entre o SIFT, o PCA-SIFT e vários outros descritores. Nesse artigo, além do comparativo, os autores propoem um descritor chamado GLOH, que é uma variação do descritor padrão do SIFT.

Visando atenuar o número de falsos *matches* por causa de variações na iluminação, em (Pirzada, Syed Jahanzeb HussainBait; HAL; SHIN, 2011) é proposta uma modificação no *threshold* do contraste. Ao invés de utilizar apenas um nível de *threshold*, são criados dois níveis de *threshold*, um *threshold* t_c para as regiões claras da imagem e outro t_e para as regiões escuras. Os *keypoints* que pertençam à regiões claras são filtrados somente pelo *threshold* t_c e os *keypoints* pertençam a regiões escuras são filtrados somente pelo *threshold* t_e. Além dessa modificação, durante a fase de *matching*, os *keypoints* que pertençam à regiões claras são comparados somente aos *keypoints* dessa mesma região. O mesmo é feito também para os *keypoints* das regiões escuras. Conforme apresentado pelos autores, essa modificação possibilita um pequeno aumento no número de *keypoints* e *matches* e redução no número de falsos *matches*.

Além dos citados, outros trabalhos tem sido realizados no intuinto de melhorar algum aspecto do SIFT, como em (GRABNER; GRAB-

NER; BISCHOF, 2006), em que é proposto o uso de imagens integrais, visando tornar mais eficiente a detecção de *keypoints* e a geração do descritor.

2.7 CONCLUSÃO

O algoritmo SURF (Speeded-Up Robust Features) (BAY et al., 2008) possibilita resultados muito semelhantes aos do SIFT. Entretanto, o SIFT gera descritores com uma localização no espaço (x,y) e na escala, mais precisa do que o SURF, bem como, é mais robusto em relação a alterações de *viewpoint* (LAGANIèRE, 2011) (KHAN; MCCANE; WYVILL, 2011).

O SIFT tem sido amplamente utilizado, em conjunto com outras técnicas, no desenvolvimento de sistemas de inspeção visual (BOHLOOL; TAGHANAKI, 2008) (YAN et al., 2009) (SHIN; MOON, 2009) (KUMAR; PRASAD; MAMATHA, 2010). Isso pode ser compreendido pelo fato de que as características como, invariância à escala, rotação e iluminação (parcialmente), bem como, robustez e distintividade do descritor, dentre outras, tornam o SIFT adequado para muitas aplicações industriais (LOCH; SZYMANSKI; STEMMER, 2013).

3 PROJETO E IMPLEMENTAÇÃO DE UM SISTEMA DE INSPEÇÃO VISUAL DE PCI

Conforme relatado no capítulo 1.2, este trabalho visa o desenvolvimento de um sistema de inspeção visual de PCI, no contexto da produção em pequenas séries. Diferentemente de outros trabalhos relacionados à inspeção de PCI, neste trabalho buscou-se desenvolver um sistema de processamento de imagens que fosse robusto e ao mesmo tempo não necessitasse de uma extensa etapa de treinamento (LIN; SU, 2006) (OPREA et al., 2007) (LONGJIANG; YING; SHENGHE, 2007) (CHO; PARK, 2008) (LETA; FELICIANO, 2008) (OLIVEIRA; PIO, 2008) (SUNDARAJ, 2009). Observa-se que, em geral, as técnicas propostas para inspecionar PCI atendem a um objetivo específico como, por exemplo, reduzir o tempo de processamento. Com isso, buscou-se neste trabalho o desenvolvimento de um sistema que pudesse reunir as diversas qualidades dos métodos já desenvolvidos. Entretanto, verificou-se que, reunir todas essas técnicas em um único sistema o tornaria muito complexo de configurar e afetaria o desempenho do mesmo. Na observância desse aspecto, percebeu-se que o algoritmo SIFT reune muitas das qualidades que estava-se objetivando reunir no mesmo sistema de inspeção. Sendo assim, optou-se por utilizar o SIFT como algoritmo central do sistema de inspeção visual desenvolvido.

Considerando que o sistema desenvolvido possui características muito diferentes dos sistemas convencionais (a citar, possui um único método principal e não possui banco de dados convencional), verificou-se que o uso rigoroso de alguma metodologia para desenvolvimento de software tornaria o processo de desenvolvimento burocrático e não contribuiria de forma significativa para o mesmo. Dessa forma, optou-se por utilizar alguns elementos de projeto de sistemas de forma que as necessidades de projeto e documentação fossem atendidas e o processo não se tornasse oneroso.

A seguir são apresentados os requisitos do sistema de inspeção visual de PCI proposto neste trabalho, as ferramentas e técnicas que foram utilizadas no desenvolvimento, bem como, a estrutura do software que foi desenvolvido.

3.1 ANÁLISE DE REQUISITOS

O levantamento de requisitos do sistema foi realizado principalmente a partir de necessidades encontradas na linha de produção do LABelectron. Outros requisitos surgiram de melhorias propostas no projeto BRAGECRIM 013/09.

Abaixo são descritos os requisitos, separados em funcionais e não-funcionais. Para cada requisito, é feita uma observação sobre a forma como o mesmo é contemplado pelo sistema desenvolvido.

Requisitos funcionais

RF-1. O quê inspecionar?
Em muitos casos, as PCI são produzidas em grandes painéis para depois serem separadas. Sendo assim, o sistema de inspeção deve ser capaz de inspecionar tanto placas individuais como painéis.
Observação: Para que o software de processamento de imagens fosse utilizado como um *plugin* e pudesse atender ao requisito NF-3, foi desenvolvido de forma que todos os parâmetros necessários sejam fornecido por um outro sistema. Nesse caso, o sistema que irá fornecer todos os dados para a análise de defeitos nas placas é o software de gerenciamento da máquina de inspeção. Sendo apenas um *plugin* para processamento de imagens, o software não possui interação com a máquina de inspeção, nem com o usuário. Considerando esses fatores e a arquitetura do software de gerenciamento da máquina, todas as imagens que serão utilizadas na análise de defeitos já estarão capturadas. Sendo assim, não haverá necessidade do software de processamento de imagens tratar de forma diferenciada as inspeções de painéis e placas individuais.

RF-2. Defeitos verificados
Os tipos de defeitos procurados pelo sistema de inspeção devem ser: componente ausente, componente rotacionado/invertido, componente deslocado e componente errado. Todos esses defeitos estão relacionados à etapa de inserção, da linha de produção.
Observação: Todos esses tipos de defeitos são detectados pelo software de processamento de imagens. O defeito em que um componente errado foi inserido na placa é detectável mas possui uma restrição. Para detectar que um componente foi trocado, o componente que foi inserido deve ter sua imagem registrada previamente no banco de dados de componentes, conforme relatado no capítulo 3.4.

RF-3. Integração com o sistema existente
Como atualmente já existe um sistema desenvolvido para controlar a S2iAOI (MELO, 2013), o sistema de processamento de imagens deve ser integrável ao sistema existente (software de gerenciamento da máquina), sem interferir no funcionamento do mesmo.
Observação: o software de processamento de imagens foi desenvolvido para ser utilizado como um *plugin*. O software foi desenvolvido utilizando a mesma linguagem de programação, o mesmo sistema operacional e a mesma biblioteca de processamento de imagens utilizados pelo sistema existente.

Requisitos não-funcionais

NF-1. Tempo de *setup*
O tempo de *setup* de uma inspeção é um fator muito importante na PPS e deve ser menor do que o tempo de *setup* para uma placa produzida em grande escala. Em uma típica PPS, o tempo de *setup* de uma AOI pode levar de 3 horas a 1,5 dia.
Observação: uma abordagem muito utilizada em sistemas de inspeção visual é a utilização de uma etapa de treinamento. Essa etapa acaba por aumentar o tempo de *setup*. Com isso, optou-se por desenvolver um sistema em que não fosse necessária uma etapa de treinamento. A etapa de *setup* restringe-se a selecionar as regiões de uma placa perfeita (*golden board*) que serão inspecionadas. Cada região selecionada corresponde a um único componente. Apesar do *setup* propriamente dito não ser realizado pelo software de processamento de imagens, esse requisito teve de ser tratado pelo mesmo. Isso ocorre pelo fato de que o *setup* de uma inspeção depende dos dados requeridos pelo software de processamento de imagens para realizar a detecção de defeitos. Sendo assim, caso o software de processamento de imagens necessitasse de mais dados além das imagens coletadas, o procedimento de *setup* teria que ser modificado.

NF-2. Tempo de inspeção
O tempo de inspeção para PPS não é um fator de grande importância. No entanto, quanto mais rápida a inspeção, melhor será. O tempo de inspeção é bastante relativo e varia conforme a densidade de componentes da placa sendo inspecionada. Para uma produção em larga escala, o tempo aproximado de inspeção por placa varia de 11 segundos a 1 minuto.

Observação: o software de processamento de imagens está centrado no algoritmo SIFT. Apesar de não ser o mais eficiente algoritmo para essa finalidade, seu desempenho é considerado adequado para aplicações industriais (LOCH; SZYMANSKI; STEMMER, 2013). Uma vez que o procedimento de inspeção requer apenas o *matching* entre os *keypoints* das imagens envolvidas, os *keypoints* das imagens da placa perfeita podem ser gerados na etapa de *setup*. Isso aumenta ainda mais a eficiência da inspeção. Outro fator que contribui para um bom desempenho da inspeção é a verificação sequencial dos defeitos. Com esse tipo de verificação, quando o software de inspeção já tem os dados necessários para concluir que um defeito foi encontrado, a inspeção para o componente em questão é encerrada, evitando verificações desnecessárias.

NF-3. Possibilidade de agentificação
O sistema de processamento de imagens deve ser desenvolvido de forma que, ao ser integrado ao software de gerenciamento da máquina de inspeção, possibilite que o sistema de inspeção como um todo possa ser implementado como um agente de software.
Observação: reiterando o que fora descrito no requisito RF-1, o software de processamento de imagens foi desenvolvido para ser utilizado como um recurso adicional (plugin) pertencente ao software de gerenciamento da máquina de inspeção. Como a agentificação do sistema de inspeção como um todo ocorrerá principalmente no software de gerenciamento da máquina, não impactará no software de processamento de imagens.

3.2 ARQUITETURA DE PROCESSAMENTO DE IMAGENS

Baseando-se no uso do SIFT, é proposta neste trabalho uma arquitetura (pipeline) de processamento de imagens. Essa arquitetura é utilizada para desenvolver um software de inspeção que, dados um banco de dados de fotos de componentes eletrônicos, um conjunto de imagens de referência e um conjunto de imagens de teste, realiza uma busca por defeitos. A figura 9 apresenta o fluxograma representativo da arquitetura proposta.

A primeira etapa da arquitetura proposta é a separação do processamento em função da presença ou não de serigrafia (ou alguma marcação) no componente. A serigrafia é uma marcação feita na superfície do componente, geralmente apresentando o código de identificação do componente. A figura 10 ilustra um exemplo de componentes

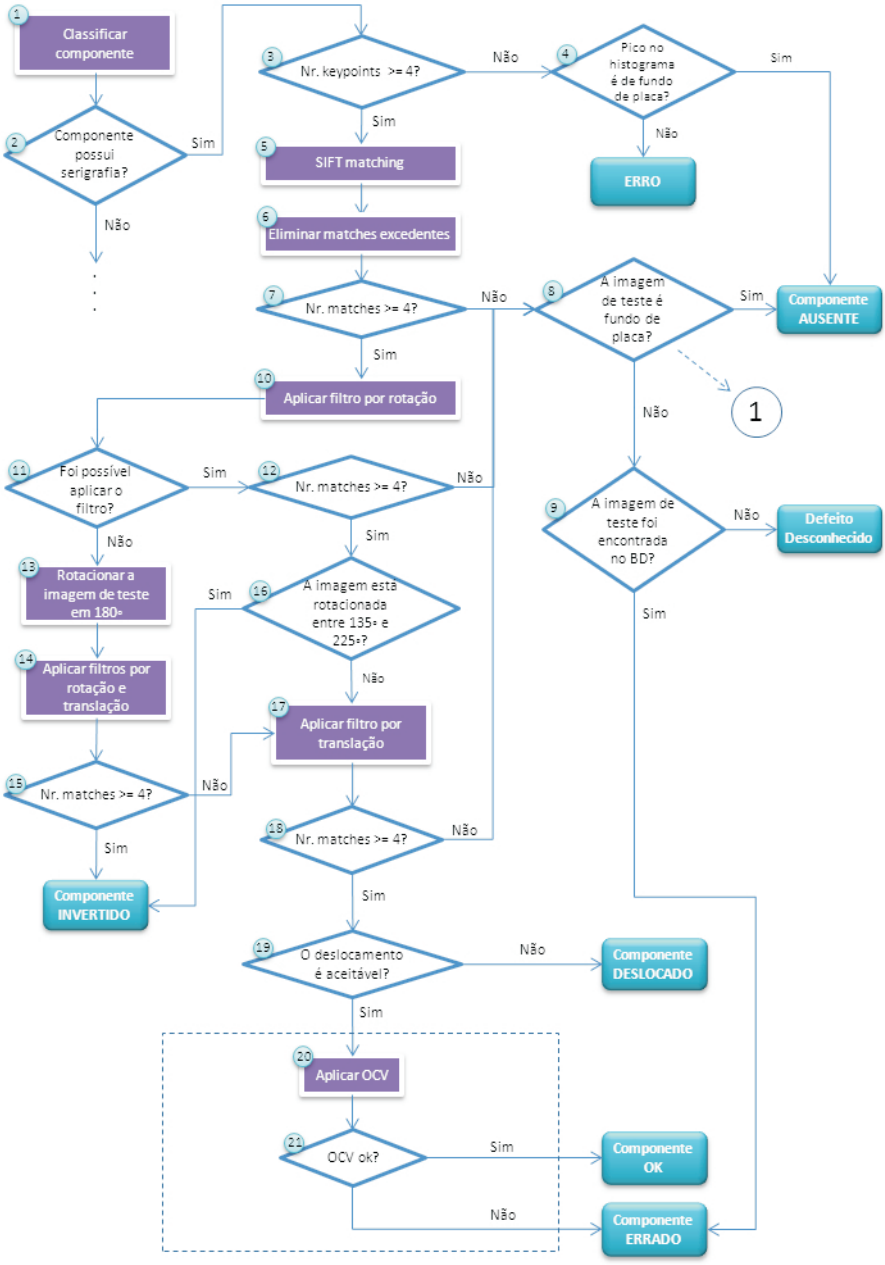


Figura 9 – Fluxograma da arquitetura de processamento de imagens proposta.

com e sem serigrafia.

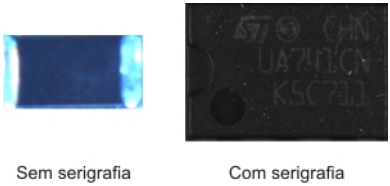


Figura 10 – Exemplo de componentes eletrônicos com e sem serigrafia.

A arquitetura proposta neste trabalho trata apenas de imagens de componentes com serigrafia. Isso se deve ao fato de que, para os componentes sem serigrafia são gerados *keypoints* SIFT de forma instável. Esses tipos de componentes são caracterizados por superfícies com cores sólidas no corpo do componente. Isso acarreta a geração de grande parte dos *keypoints* na área de contato entre o corpo do componente e os eletrodos pois, essa é justamente a área com maior variação de contraste nesses tipos de componente. A figura 11 contém fotos de componentes sem serigrafia, juntamente com a localização dos *keypoints* gerados com essas fotos. Como pode ser observado na imagem "A", existe uma concentração maior de *keypoints* próximos aos eletrodos. Na imagem "B" nota-se também a presença de *keypoints* no corpo do componente e em regiões dos eletrodos que sofreram variações em função de imperfeições na superfície. Essas imperfeições ocasionam alterações na luz refletida e consequentemente, na intensidade dos *pixels*. Teoricamente, o corpo desses tipos de componentes possuem a superfície lisa e uma cor constante. Entretanto, na prática, ocorre o surgimento de imperfeições, ocasionando a geração de *keypoints*. O problema reside no fato de que esses *keypoints* são instáveis, tanto em relação aos valores do descritor, como em relação à localização. Portanto, conclui-se que os *keypoints* gerados a partir de fotos desses tipos de componentes não são adequados para realizar inspeções.

A classificação dos componentes em função da existência ou não de serigrafia não foi implementada nesse trabalho. Com isso, durante a programação da inspeção o usuário fará essa diferenciação, ou seja, somente serão criadas janelas de inspeção para componentes com serigrafia.

Após essa classificação, inicia-se o processo de inspeção baseado no SIFT. Primeiramente, verifica-se a quantidade de *keypoints* gerados a partir das imagens. Caso a quantidade de *keypoints* seja inferior ao

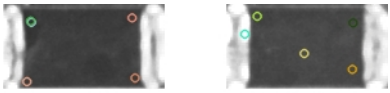


Figura 11 – *Keypoints* gerados a partir de fotos de componentes eletrônicos SMD sem serigrafia.

número mínimo aceitável de *matches* (4), verifica-se o pico do histograma da imagem (bloco 4). Caso o pico do histograma seja correspondente ao pico do fundo da placa, considera-se que o componente está ausente, caso contrário, é impossível realizar a inspeção. Se a quantidade de *keypoints* for igual ou superior ao mínimo aceitável, a inspeção continua. Obviamente, a quantidade de *keypoints* da imagem de referência pode ser checada no momento do *setup*.

Tendo um número suficiente de *keypoints*, efetua-se o *matching* utilizando a imagem de referência (*gold*) do componente sendo inspecionado e a imagem de teste do mesmo (bloco 5). Lowe sugere que, após um procedimento de *matching*, sejam eliminados os *matches* cuja razão entre o mais próximo e o segundo mais próximo *keypoint* seja superior a 0,8 (LOWE, 2004), conforme apresentado no capítulo 2.6. Todavia, nos experimentos realizados neste trabalho, esse filtro não foi utilizado, para que se pudesse manter a quantidade de bons *matches*. Lowe afirma que uma das razões para que a busca aproximada funcione bem é que, levando-se em consideração os resultados do filtro acima citado, não há a necessidade de se procurar exatamente o *keypoint* com o descritor mais próximo. Observou-se no entanto que, mesmo não utilizando o filtro, os *matchings* baseados em uma busca aproximada tenderam a gerar resultados muito similares aos da busca por força bruta, bem como, reduzir o tempo da busca.

Após a obtenção dos *matches*, é necessário eliminar aqueles excedentes, ou seja, os *matches* em que um dos *keypoints* está presente em algum outro *match* (bloco 6). Essa eliminação é importante porque a quantidade de *matches* é utilizada para determinar se um componente eletrônico está presente na imagem. O critério utilizado para eliminar os *matches* excedentes é a semelhança entre os *keypoints*, ou seja, serão mantidos os *matches* cujos *keypoints* forem mais semelhantes. Para determinar essa semelhança, calcula-se a distância euclidiana entre os dados "x, y, θ" (translação e rotação) dos *keypoints* originários da imagem de referência e da imagem de teste, associados ao respec-

tivo *match*. Nesse processo surge a situação em que um determinado *keypoint* pode ser mais similar em um dado *match*, mas também pode ser ainda mais similar em um outro *match* a que tenha sido associado. Com isso, torna-se necessário verificar a similaridade de cada *keypoint* em todos os *matches* a que esteja vinculado. No intuito de evitar que fossem efetuadas sucessidas verificações de similaridade de um mesmo *keypoint*, encontrou-se uma forma simples de resolver esse problema, ordenando por similaridade o vetor de *matches*. Essa ordenação por similaridade garante que sempre serão selecionados os *matches* mais similares para cada par de *keypoints*. O procedimento para eliminação de *matches* excedentes é apresentado no algoritmo 1.

Algoritmo 1: Eliminar matches excedentes

Entrada: Match[ ] matchesInicial: Conjunto de todos os matches obtidos.
Saída: Match[ ] matchesResultantes: Conjunto de matches sem excedentes.

```
1  início
2      Match[ ] matchesResultantes = ∅;
3      Keypoint[ ] arrayKptsRef = ∅;
4      Keypoint[ ] arrayKptsTst = ∅;
5      ordenaCrescentementePorSimilaridade(matchesInicial);
6      para cada Match m ∈ matchesInicial faça
7          Keypoint kptRef = extraiKeypointReferencia(m);
8          Keypoint kptTst = extraiKeypointTeste(m);
9          if (kptRef ∉ arrayKptsRef) e (kptTst ∉
           arrayKptsTst) then
10             adiciona(kptRef, arrayKptsRef);
11             adiciona(kptTst, arrayKptsTst);
12             adiciona(m, matchesResultantes);
13         end
14     fim
15     retorna matchesResultantes;
16 fim
```

O algoritmo 1 recebe como entrada o conjunto total de *matches* obtidos com os *keypoints* SIFT. Para evitar que um mesmo *keypoint* esteja vinculado a mais do que um *match*, são utilizados dois vetores, que armazenam os *keypoints* que já foram vinculados a algum *match*. Inicialmente esses vetores estão vazios, bem como, a listagem dos *matches* resultantes (linhas 2 a 4). Após isso, o vetor inicial de *matches* é ordenado crescentemente por similaridade (linha 5), conforme expli-

cado anteriormente. Então, percorrendo-se todos os *matches* (linha 6), serão adicionados na lista de *matches* resultantes somente aqueles em que nenhum dos *keypoints* envolvidos esteja vinculado a algum *match* já adicionado na lista resultante (linhas 7 a 13). Nas linhas 7 e 8, são apenas obtidos os respectivos ponteiros para os *keypoints* envolvidos no *match* em questão.

Eliminados os *matches* excedentes, verifica-se a quantidade de *matches* resultante (bloco 7). Lowe sugere que, com 3 bons *matches* é possível realizar um reconhecimento confiável de objetos (LOWE, 2004). No entanto, verificou-se que, para certas fotos obtidas do fundo da placa (fotos de referência ou de componentes ausentes), surge uma grande quantidade de *keypoints*. Em muitos casos, apesar de aleatórios, esses *keypoints* originam *matches* que acabam por passar pelos filtros de falsos *matches*. Sendo assim, se a quantidade de *matches* for inferior a 4, após a eliminação dos *matches* excedentes, será considerado que o componente não está presente na placa. A alteração da quantidade mínima de *matches* de 3 para 4 fez com que nenhuma foto de componente fosse associada ao fundo da placa, bem como, reduziu significativamente a associação equivocada de fotos de componentes. Além do mais, como a janela de inspeção é delimitada para abranger apenas o corpo do componente, evitando os terminais, reduz-se a possibilidade do surgimento de *keypoints* espúrios, que ocasionem faltos *matches*. Caso o número de *matches* for maior ou igual a 4, o processamento continua, caso contrário, é verificado se a foto se trata de um fundo de placa. Sempre que uma foto for considerada fundo de placa, conclui-se que o componente está ausente. Esse procedimento é descrito em mais detalhes adiante. Se uma imagem não for considerada fundo de placa (bloco 8), é verificado se a imagem de teste foi encontrada no banco de dados de componentes (bloco 9). Caso tenha sido encontrada, conclui-se que o componente está trocado na placa. Caso contrário, o defeito é desconhecido.

Tendo concluído que o componente, ou parte dele, está presente na foto, aplica-se um filtro por rotação. Esse filtro, descrito em mais detalhes no capítulo 4.4, resulta na eliminação de falsos *matches*, bem como, na obtenção do ângulo de rotação de um componente na placa. Caso esse filtro não possa ser aplicado, a imagem será rotacionada em 180° na tentativa de verificar se o componente está invertido. Após essa rotação e a aplicação do filtro por rotação novamente e de um filtro de falsos *matches* por translação (descrito em mais detalhes no capítulo 4.5), se o número de *matches* resultante for maior ou igual a 4, conclui-se que o componente está invertido. Caso contrário, o processa-

mento continua a partir da verificação do deslocamento do componente (bloco 17), utilizando a imagem de teste original (não rotacionada). Sendo possível aplicar o filtro por rotação, verifica-se a quantidade de *matches* resultante. Sendo inferior a 4, verifica-se novamente se a imagem de teste é fundo de placa. Sendo maior ou igual a 4, verifica-se o ângulo de rotação do componente. Se o ângulo de rotação estiver entre 135° e 225°, conclui-se que o componente está invertido. Caso contrário, o processo continua a partir da verificação do deslocamento do componente (bloco 17).

Para verificar se um componente está deslocado na placa, aplica-se um filtro de falsos *matches* por translação. Esse filtro, além de eliminar falsos *matches*, possibilita obter o deslocamento do componente nos eixos X e Y. Se o deslocamento estiver fora de uma faixa aceitável, configurada previamente, o componente será considerado deslocado. Caso contrário, o processo é encerrado e conclui-se que o componente está corretamente posicionado na placa.

Mesmo com um número suficiente de *matches* indicando que o componente está presente e posicionado corretamente na placa, é importante uma última verificação. Uma das vantagens de se utilizar o SIFT na detecção de objetos é a possibilidade de efetuá-la, mesmo na presença de oclusão parcial. Porém, no caso da verificação de componentes com serigrafia, pode ocorrer a situação em que um componente errado tenha sido inserido na placa. Caso esse componente tenha parte de sua superfície semelhante a do componente de referência, surgirão bons *matches*, com *keypoints* localizados na parte semelhante da serigrafia, como se tivesse ocorrido uma oclusão parcial do componente. Isso pode ocorrer, por exemplo, nos casos em que ambos os componentes (o correto e o errado) tenham alguns caracteres semelhantes na serigrafia. Diante dessa situação, sugere-se uma verificação da serigrafia utilizando alguma outra técnica. Dentre as técnicas pesquisadas, observou-se que duas podem ser úteis na solução dessa deficiência: verificação de caracteres e *image hashing*. A verificação de caracteres pode ser feita com uso de OCR (Optical Character Recognition), utilizada em inspeção de PCI há bastante tempo (HATA et al., 1989), ou OCV (Optical Character Verification). O uso de OCR torna o processo mais confiável pois, sabe-se exatamente os caracteres que foram identificados na superfície do componente. Entretanto, essa técnica é sensível a deformações na serigrafia dos componentes, podendo ocasionar uma alta taxa de falsos negativos. Além do mais, geralmente utiliza-se redes neurais para realizar a identificação dos componentes, o que torna necessária a criação de uma grande base de dados de componentes

para que se tenha inspeções mais robustas (WANG; LI; LUO, 2003)(OH; JUNG; PARK, 2004). Esse fato tende a onerar o processo de *setup*. Acredita-se que o uso de OCV ao invés de OCR, possa ser aplicado de forma a encontrar-se um equilíbrio entre confiabilidade e robustez. A outra alternativa com potencial para auxiliar na verificação da serigrafia é *image hashing*. Essa técnica é tradicionalmente utilizada para gerar assinaturas de imagens (SCHNEIDER; CHANG, 1996)(LEFEBVRE; CZYZ; MACQ, 2003). *Image hashing* diferencia-se das técnicas tradicionais de hash (como as utilizadas em criptografia) por ser baseada no conteúdo da imagem e não fortemente vinculada aos bits. Isso significa que o *image hashing* é robusto o suficiente para ser afetado apenas pelo conteúdo percebido na imagem e não por pequenas modificações nos *pixels*. Em (LONGJIANG; YING; SHENGHE, 2007) é apresentada uma forma de uso do *image hashing* na inspeção da serigrafia de PCI. Os resultados apresentados, apesar de serem baseados em poucos testes, são aparentemente promissores.

Um procedimento muito utilizado em todo o processo é a verificação se a imagem de teste refere-se a um fundo de placa ou a algum componente inserido equivocadamente. Esse procedimento está representado no fluxograma da figura 12. Basicamente, uma foto será considerada fundo de placa se não for encontrada no banco de dados de componentes e se o pico no histograma corresponder ao pico no histograma da foto de referência do fundo da placa. Primeiramente é procurada alguma correspondência da imagem de teste no banco de dados (blocos 1 e 2). Se não for encontrada, a imagem será rotacionada em 180° e procurada novamente no banco de dados (blocos 3, 4 e 5). A imagem precisa ser rotacionada nesta etapa pela mesma razão com que pode ser rotacionada na verificação do ângulo de rotação do componente.

Para considerar que o pico de um histograma corresponde a um fundo de placa, verifica-se o *pixel* correspondente ao maior pico, que seja inferior a 250, em escala de cinza. Os *pixels* na faixa de 250 a 255 geralmente caracterizam desenhos de contorno de componentes ou escritas, impressos no fundo da placa. Como a serigrafia dos componentes pode também estar nessa mesma faixa de valores de *pixel*, evita-se o uso da mesma. A verificação do histograma torna o processo mais robusto, uma vez que, mesmo não sendo fundo de placa, uma determinda foto pode não ter sido encontrada no banco de dados. Isso pode ocorrer por não ter sido gerado um número suficiente de *matches* durante a busca do componente, ou até mesmo, o componente inserido equivocadamente na placa não estar presente no banco de dados. A detecção

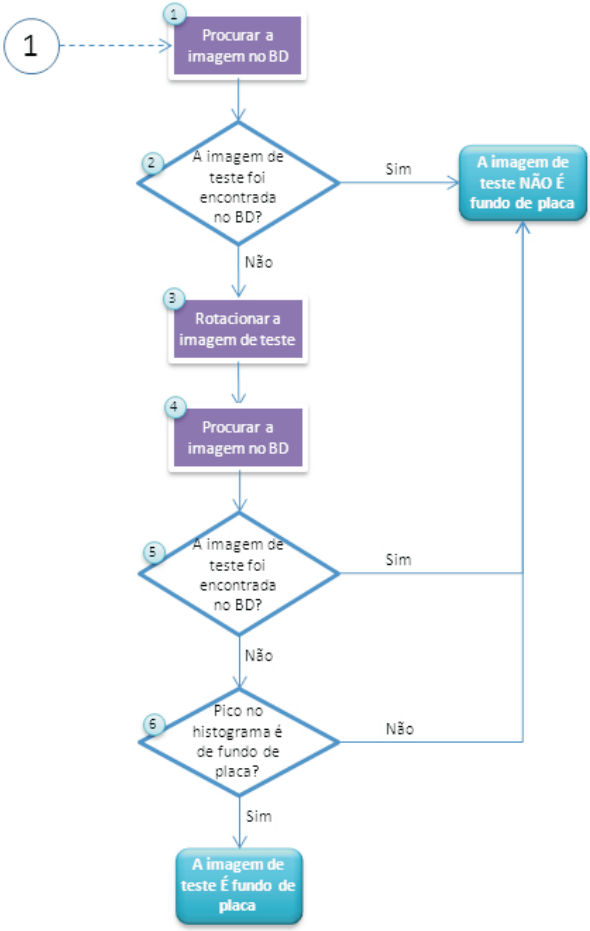


Figura 12 – Fluxograma representativo do processo que verifica se uma determinada imagem se refere ao fundo da placa ou a algum componente

do maior pico no histograma da foto de referência do fundo da placa é feito da mesma forma que é feita a detecção para a foto de teste, comentado acima. Visando tornar o processo mais robusto, o pico P_r da foto de teste corresponderá ao pico P_f do fundo da placa caso tenha uma diferença de no máximo 5%, ou seja, se $\frac{|P_r - P_f|}{256} \leq 0,05$.

Uma deficiência evidente dessa verificação é que o componente inserido na placa pode não estar presente no banco de dados e também possuir uma faixa de cores, em escala de cinza, que corresponda ao fundo da placa. Na tentativa de contornar esse problema, foram testadas diversas técnicas, sem sucesso. Todas as técnicas foram testadas no intuito de identificar, sem nenhuma etapa de treinamento, se uma determinada imagem possui características suficientes para classificá-la como fundo de placa. As técnicas testadas foram as seguintes:

- Média aritmética em escala de cinza: buscou-se através da média aritmética do valor dos *pixels*, em escala de cinza, descobrir algum valor que caracterizasse o fundo da placa. Em geral, apresentou bons resultados, mas não é robusta. Determinadas imagens de componentes podem ter *pixels*, cuja média aritmética seja igual ou próxima à média aritmética do fundo da placa;

- Quantidade de *pixels* pertencentes ao fundo da placa: utilizando a média aritmética conforme descrito acima, calcula-se a quantidade de *pixels* que pertençam ao fundo da placa. Para isso, utiliza-se uma margem de tolerância. Por exemplo, seja VM o valor da média aritmética do fundo da placa e VP o valor de um *pixel* em escala de cinza, são considerados pertencentes ao fundo da placa, os *pixels* em que "$VM - 20\% \geq VP \leq VM + 20\%$";

- Menor valor, média aritmética e maior valor, para cada canal RGB: analisando separadamente os canais RGB de fotos de fundo de placa, procurou-se algum padrão que se destacasse. Porém, nenhum canal apresentou um padrão específico para fundo de placa;

- Média aritmética dos canais HSV: convertendo as imagens para o formato HSV, houve a tentativa de buscar algum padrão utilizando a média aritmética de cada canal individualmente;

- Busca por picos no histograma em escala de cinza: após gerar o histograma da imagem do fundo da placa, os maiores picos (valores que receberam mais votos) eram detectados. Após isso, para cada imagem de teste calcula-se o histograma e os picos são detectados. Por último, verifica-se a presença dos picos da imagem do fundo da placa dentre os picos da imagem de teste. Por exemplo, procura-se os 3 maiores picos do fundo da placa entre os 5 maiores picos da imagem de teste. Essa abordagem foi testada no intuito de tornar mais robusta a verificação de um único pico, esta última, utilizada como complemento no final do fluxo (bloco 6) apresentado na figura 12;

- Busca por agrupamentos de picos no histograma em escala de cinza: igualmente ao descrito acima, são procurados os picos nos histogramas, porém, os picos próximos são agrupados. Dessa forma, ao invés de se procurar por valores específicos entre os picos do histograma da imagem de teste, procura-se pelos picos que estejam dentro da faixa de valores dos picos agrupados. Por exemplo, suponha-se que tenham sido encontrados dois picos no histograma do fundo da placa, um pico P_1, que compreende valores na faixa de 65 a 72 e um pico P_2, que compreende valores na faixa de 232 a 237. Sendo assim, dentre os picos da imagem de teste, serão procurados aqueles que estejam dentro da faixa de P_1 ou de P_2.

- Utilização de *features* Haralick: uma das *features* mais utilizadas na classificação de imagens com texturas são as chamadas *Haralick features* (HARALICK; SHANMUGAM; DINSTEIN, 1973). Na tentativa de se encontrar alguma característica na textura das fotos de fundo de placa, foram testadas 11 (*features* 1 à 11) das 14 *features* Haralick. Buscou-se encontrar alguma *feature* que pudesse melhor representar o fundo da placa e portanto, fornecer algum *threshold* genérico para a placa em questão;

- Distância euclidiana utilizando a média aritmética dos canais RGB: considerando que, individualmente os canais RGB não permitem caracterizar um fundo de placa, foi testada a distância euclidiana entre a média aritmética de cada canal RGB. Isso foi feito no intuito de detectar alguma relação entre os canais RGB, que pudesse caracterizar o fundo da placa;

- Cálculo de SAD entre o valor médio de cada canal RGB: calcula-se a média aritmética de cada canal RGB. Utilizando essas três médias, calcula-se o valor da SAD (Sum of Absolute Differences) entre os valores obtidos a partir da imagem do fundo da placa e da imagem de teste.

Conforme comentado no capítulo 2.6, o procedimento para o reconhecimento de objetos usando o SIFT, sugerido por Lowe, inicia-se por verificar a razão entre o *keypoint* mais próximo e o segundo mais próximo, após a realização do *matching* (LOWE, 2004). Essa verificação não foi necessária, conforme comentado anteriormente. A etapa seguinte seria então, realizar a clusterização dos *keypoints*, baseada na transformada generalizada de Hough. No entanto, realizando testes com o uso da transformada de Hough, conforme sugerido por Lowe, a imensa quantidade de *matches* é eliminada, inviabilizando o sistema.

Aplicando uma filtragem baseada na transformada de Hough aos *matches* da figura 16, nenhum *match* restou. Diversos *clusters* foram formados e foi detectado apenas um pico no espaço de Hough, com 2 votos. Levando-se em conta que, para um *match* confiável, são necessários pelo menos 3 votos, a filtragem com o uso da transformada de Hough impossibilitou a utilização dos *matches* obtidos com as imagens.

Com esses resultados, torna-se desnecessário prosseguir com as etapas para reconhecimento de objetos, sugeridas por Lowe. Por esse motivo, foram utilizados os algoritmos apresentados no capítulo 4.

Um fator que contribui para esses resultados é a clusterização baseada nos dados de translação, rotação e escala. Isso torna o filtro mais restritivo, visto que, um dado *match* precisa coincidir nessas três dimensões para que possa passar pelo filtro. Considerando que na inspeção de PCI há pouca ou nenhuma variação de escala, essa dimensão pode ser eliminada dos filtros. Nos experimentos realizados verificou-se que a eliminação dessa dimensão nos filtros de falsos *matches* não prejudica a precisão dos resultados, enquanto que, possibilita que um maior número de *matches* passe pelo filtro. Dentre os *keypoints* obtidos com o uso das fotos da figura 14 percebeu-se uma considerável variação na escala em que os *keypoints* foram detectados. Também foram observadas variações de escala entre os respectivos *keypoints* envolvidos nos *matches* apresentados na figura 16. Analisando esses fatores, conclui-se que, para aplicações em que a variação de escala seja conhecida ou inexistente, não há a necessidade de se utilizar a escala em que os *keypoints* são detectados.

3.3 FERRAMENTAS E TÉCNICAS UTILIZADAS

O software de processamento de imagens foi desenvolvido baseado na versão 2.4.0 da biblioteca de processamento de imagens OpenCV (TEAM, 2013). Dentre as muitas vantagens dessa biblioteca, destaca-se a reunião de implementações de uma elevada quantidade de algoritmos, incluindo o SIFT, atualizações contínuas (incluindo melhorias nos algoritmos já implementados) e interface de comunicação com dispositivos periféricos. Somando-se a essas características o fato de que já foi utilizada pelo software de gerenciamento da máquina para captura de imagens, optou-se por utilizar essa biblioteca também no desenvolvimento do software de processamento de imagens.

O desenvolvimento do software de processamento de imagens foi feito totalmente em C++, rodando em Linux Ubuntu 12.04. Como

o software de gerenciamento da máquina já opera em Linux Ubuntu, optou-se por utilizar o mesmo sistema operacional, até porque, dessa forma também não haveria custo adicional com aquisição de licença de software.

Conforme citado anteriormente, o software de inspeção desenvolvido está fortemente baseado no algoritmo SIFT. Utilizando esse algoritmo, percebeu-se que é possível reduzir o tempo de *setup* das inspeções. Isso ocorre devido à estabilidade do algoritmo. Por exemplo, não é necessário grandes cuidados com variações na iluminação, bem como, imprecisão na localização física da placa sendo inspecionada. Variações de localização nos eixos X e Y serão inicialmente tratadas pelo software de gerenciamento da máquina com o algoritmo "Fiducial Finder" (MELO, 2013). Com esse algoritmo, as imagens capturadas poderão ser referenciadas em relação à posição zero (0,0) da placa. Qualquer imprecisão no posicionamento da placa que por ventura tenham passado por esse algoritmo, aparecerão como pequenos deslocamentos nos componentes e não afetarão o cálculo do descritor dos *keypoints* SIFT de forma significativa. Pequenas variações no eixo Z também não afetarão a eficácia da inspeção pois, isso já é tratado pelo SIFT com a geração do espaço de escalas. Através do SIFT, é possível realizar o *matching* entre imagens com grandes variações de escala.

3.4 ESTRUTURA DO SOFTWARE DESENVOLVIDO

O software desenvolvido para validar a proposta deste trabalho possui uma estrutura bastante simples em termos de implementação. Como não há classes no sistema, são descritos os arquivos criados. Todo o software está contemplado em 4 arquivos principais (além dos arquivos auxiliares e bibliotecas utilizadas), com as seguintes funcionalidades:

- **functions.cpp**: contém funções utilitárias do sistema tais como, rotação de imagens, cálculo do valor médio dos *pixels* de uma imagem, filtros de falsos *matches*, etc;
- **InspectionConstants.hpp**: contém a definição de algumas estruturas de dados utilizadas pelo sistema, bem como, constantes definidas para parametrizar o sistema;
- **mysift.cpp**: contém funções criadas para realizar a extração de *keypoints* SIFT a partir de imagens, bem como, realizar *matchings*

entre imagens. Foram utilizados dois tipos de implementação de *matchings*: a) baseando-se em força bruta, em que os *matches* eram formados entre os *keypoints* com a menor norma L_2 (distância euclidiana) entre si; b) baseando-se na busca aproximada do vizinho mais próximo, cuja implementação utiliza a biblioteca FLANN (Fast Library for Approximate Nearest Neighbors) (MUJA; LOWE, 2009), distribuída juntamente com o OpenCV. Essa biblioteca contém a implementação de diferentes algoritmos para busca do vizinho mais próximo. A seleção do algoritmo mais adequado, bem como, dos parâmetros ótimos para o mesmo, é feita automaticamente, em função do conjunto de dados de entrada (no caso do SIFT, os descritores dos *keypoints*) (MUJA; LOWE, 2013).

- **InspectionSystem.cpp**: arquivo principal do software desenvolvido. Neste arquivo está contida a implementação da arquitetura de processamento de imagens proposta neste trabalho.

A execução do software é feita a partir do método "main" do arquivo "InspectionSystem.cpp", passando como parâmetro a pasta que será inspecionada. Dentro dessa pasta, deverão estar contidas as imagens adquiridas da placa de teste, as respectivas imagens de referência adquiridas da placa perfeita (*golden board*) e o banco de dados de componentes. Os arquivos de imagens dentro da pasta da inspeção estão divididos em casos de teste. Cada caso de teste possui uma imagem de referência do fundo da placa, uma imagem de referência do componente perfeito, e N testes. Cada um dos N testes representa uma imagem adquirida na posição do componente na placa e pode apresentar um característica diferente como, por exemplo, deslocamento do componente, componente ausente, componente perfeito, variação na iluminação, etc. Os arquivos de cada caso de teste devem obedecer o seguinte padrão:

NR_CASO_TESTE_**BACKGROUND:**
Imagem de referência do fundo da placa. A imagem não precisa ser adquirida necessariamente na mesma posição (região) do componente.
NR_CASO_TESTE_**GOLD:**
Imagem de referência do componente na placa perfeita.
NR_CASO_TESTE_**TEST**_SEQ_TESTE:
Teste específico.

Exemplos:

1_BACKGROUND.png
1_GOLD.png
1_TEST_1.png
1_TEST_2.png
1_TEST_3.png
2_BACKGROUND.png
2_GOLD.png
2_TEST_1.png
2_TEST_2.png
2_TEST_3.png

Dentro da pasta da inspeção deve haver uma pasta nomeada "bd" com as imagens que formam o banco de dados de componentes. Cada imagem do banco de dados deve seguir o padrão "NR_NOME". Por exemplo:
1_CI039Y817A.png
2_RES823K.png
3_72C6DNM.png

Ao ser executado, o software percorre todos os casos de teste e para cada teste é executado o procedimento de inspeção descrito na seção 3.2.

3.5 CONCLUSÃO

O padrão esperado para as imagens de uma inspeção e para o banco de dados de componentes não é exatamente o padrão utilizado pelo software de gerenciamento da máquina. Isso foi feito dessa forma pelos seguintes motivos:

- Facilitar os testes do software de inspeção;
- Evitar que possíveis problemas de integração entre os softwares interferissem nos resultados das inspeções;
- O software de gerenciamento precisará ser alterado para que a área de aquisição das imagens possa ser ajustada pelo usuário. Essa área é também conhecida como janela de inspeção. Atualmente o software de gerenciamento sempre adquire imagens com as dimensões de 640x480 *pixels*.

4 ELIMINAÇÃO DE FALSOS MATCHES BASEADA NA COERÊNCIA DE ROTAÇÃO E TRANSLAÇÃO

O *template matching* possui grande relevância em diversas aplicações envolvendo processamento de imagens. Um dos principais problemas inerentes ao mesmo é o surgimento dos falsos *matches*. As técnicas que até então têm sido utilizadas para esse fim, baseiam-se quase que unicamente em informações de um único *match* para determinar se o mesmo é correto ou não. Neste trabalho é proposta uma nova forma de filtragem de *matches*, obtidos a partir do algoritmo SIFT. A filtragem proposta baseia-se na coerência entre *matches*, com o desenvolvimento de dois algoritmos. Para tanto, são analisados dois tipos de coerência: rotação e translação. O algoritmo de filtragem por rotação pode ser utilizado em qualquer imagem e sua utilidade estende-se além da filtragem de *matches*, servindo também para detectar a inclinação de um objeto em uma cena. O algoritmo de filtragem por translação possui mais eficácia em imagens que não sofreram variação de escala ou rotação e dentro dessas condições, pode ser utilizado também para detectar o deslocamento de objetos em uma cena. Os resultados experimentais demonstram a viabilidade prática desses algoritmos, especialmente no que se refere à aplicação em ambientes controlados, como, por exemplo, inspeções industriais.

Todos *keypoints* utilizados nos testes de ambos os algoritmos, foram extraídos com o uso da implementação do SIFT distribuída junto ao OpenCV 2.4.0. Os parâmetros do SIFT foram mantidos com o valor padrão, ou seja:

- Número de camadas por oitava: 3;
- *Threshold* de contraste: 0,04;
- *Threshold* de borda (*keypoints* localizados em borda): 10;
- σ da função gaussiana: 1,6.

4.1 FALSOS MATCHES

O processo de *matching* que utiliza *keypoints* baseia-se em alguma métrica de semelhança entre os conjuntos de *keypoints* da imagem *template* e da imagem de teste. Considerando que os descritores dos *keypoints* são vetores, uma métrica muito utilizada nesse caso é

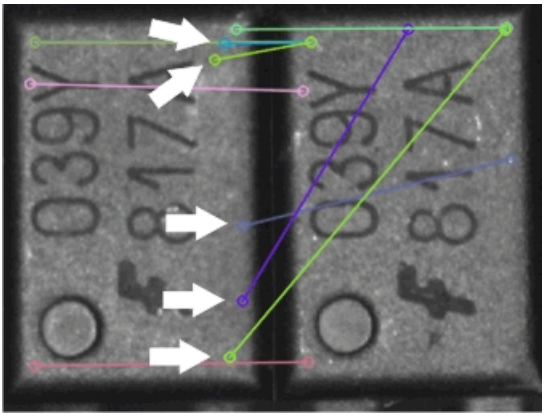


Figura 13 – Ocorrência de falsos matches

a norma euclidiana. Utilizando um algoritmo de *matching* por aproximação (BEIS; LOWE, 1997), ou mesmo utilizando força bruta, após o processo de *matching* muitas vezes ocorrem os falsos *matches*, ou seja, num dado *match*, dois *keypoints* são identificados como semelhantes, quando na verdade não são. Isso induz a erros de análise, podendo-se, por exemplo, concluir que um objeto está presente em uma imagem, quando na realidade não está, ou está em um local diferente da imagem. A figura 13 ilustra uma situação de ocorrência de falsos *matches*. Nessa figura, estão contidas as imagens *template* (à esquerda) e de teste (à direita), de dois componentes eletrônicos semelhantes. Os *matches* estão representados por linhas ligando os *keypoints* da imagem de *template* aos da imagem de teste. Os falsos *matches* estão indicados pelas setas. Note que nesta imagem surgiram 5 falsos *matches* e apenas 4 bons *matches*. Em se tratando do algoritmo SIFT, um fator que contribui significativamente para o surgimento de falsos *matches* é a utilização de imagens com baixo contraste, sendo que, nesses casos há maior probabilidade de surgirem *keypoints* com descritores semelhantes.

4.2 TRABALHOS RELACIONADOS

Wu e Dai realizaram um trabalho sobre filtragem de *matches* baseados em características (WU; DAI, 2005). Em seu trabalho, o *matching* é realizado com base na detecção de cantos (corners). Para eliminar os

falsos *matches*, é utilizada unicamente a restrição epipolar entre as duas imagens envolvidas. Para adotar essa abordagem é necessário o conhecimento dos parâmetros da restrição epipolar entre as imagens. No entanto, esses parâmetros nem sempre são conhecidos, principalmente no caso de imagens obtidas com apenas uma câmera.

Shin e Moon utilizaram o SIFT para verificar se determinadas regiões de uma imagem são suscetíveis ao surgimento de falsos *matches* (SHIN; MOON, 2009). O problema desta abordagem é que o *matching* em si é realizado com o uso do algoritmo MAD (Mean of Absolute Differences) modificado, que pertence à classe dos *matchings* baseados em regiões. Dessa forma, o *matching* é sensível a variações de escala, rotação e iluminação.

Yang, Liu e Zhang desenvolveram um trabalho que permite filtrar falsos *matches*, baseando-se no SIFT (YANG; LIU; ZHANG, 2010). No entanto, o processo de *matching* precisa ser alterado, sendo apresentado um algoritmo denominado *matching* regional. Nesse algoritmo, os autores utilizam um coeficiente de correlação cruzada para estipular um nível de qualidade para os *matches*. Para o cálculo desse coeficiente são utilizadas regiões de vizinhança dos *keypoints* envolvidos nos *matches*, baseando-se nos conceitos de geometria epipolar. No entanto, não é apresentado um estudo sobre o limiar desse coeficiente que poderá determinar um bom *match*. Um problema dessa abordagem é que a pequena região de 3x3 *pixels*, ao redor dos *pixels* centrais pode ser semelhante, induzindo a considerar um falso *match* como sendo bom. Outro problema dessa abordagem é também utilizar restrição epipolar, que possui a limitação anteriormente descrita.

Li baseia seu trabalho no SIFT e também utiliza um processo semelhante ao do *matching* regional, citado anteriormente (LI, 2011). No entanto, não é utilizada restrição epipolar e a qualidade de um *match* é determinada pela quantidade de *sub-matches* obtidos a partir do *match* principal. Uma vez que se tenha um *match* obtido com o SIFT, verifica-se, de uma forma geral, a quantidade de *matches* obtidos entre os *pixels* da região de vizinhança dos *keypoints* envolvidos no *match*. Quanto mais *matches* nas regiões, melhor a qualidade do *match* principal. Muitas vezes um *match* ruim é caracterizado pela má localização dos *keypoints*. Sendo assim, o principal problema da abordagem de Li é que a região próxima aos *keypoints* pode ser semelhante, mesmo que os *keypoints* estejam localizados em pontos distintos das imagens. A figura 13 apresenta vários exemplos de falsos *matches*, entre *keypoints* localizados em regiões distintas das imagens.

4.3 COERÊNCIA ENTRE MATCHES

A presença dos falsos *matches* dificulta a análise de imagens, principalmente no que se refere à localização de objetos. Suponha que na figura 13, deseja-se comparar se os objetos estão na mesma posição. Utilizando as informações geradas pelos *matches* resultantes, isso é impossível pois, a maior parte das informações de posição são aleatórias, ou seja, há mais falsos *matches* do que bons *matches*. No entanto, percebe-se que, enquanto os falsos *matches* são aleatórios, os bons *matches* possuem uma certa coerência. Neste trabalho, foram estudadas as coerências que se apresentam termos de translação e rotação entre os *keypoints*. Para facilitar a compreensão e visualização, os algoritmos desenvolvidos foram desmembrados em algoritmos menores, todos apresentados nas seções seguintes.

Apesar de os algoritmos terem sido desenvolvidos para aplicação em uma máquina de inspeção, verificou-se que sua abrangência pode ser estendida. Os resultados obtidos com os filtros são apresentados com a utilização de fotos capturadas em um ambiente de escritório, com uma câmera de celular e somente a iluminação disponível no ambiente. Isso foi feito para demonstrar que, apesar de limitada, a aplicabilidade dos algoritmos não está restrita a um ambiente industrial e controlado. A figura 14 apresenta as fotos utilizadas nos testes. Na figura 15 são apresentados os *matches* obtidos com as fotos A e B da figura 14. Considera-se que a foto A representa uma imagem *template* e a foto B uma imagem de teste. O *matching* foi realizado com base em uma implementação da FLANN, conforme comentado no capítulo 3.4.

Como pode ser observado, em função da quantidade de *matches* obtidos, torna-se difícil analisá-los visualmente. Então, para facilitar a visualização dos *matches* e do resultado da aplicação dos algoritmos, aplicou-se um filtro nos *matches* da figura 15. Seja T o conjunto de todos os *matches* e K_1 e K_2 os conjuntos de *keypoints* obtidos a partir de duas fotos A e B, respectivamente. Para cada *match* $m_i \in T$ tem-se um *keypoint* $k_1 \in K_1$ e um *keypoint* $k_2 \in K_2$, bem como, a distância euclidiana d entre os descritores de k_1 e k_2. A menor distância euclidiana entre todos os *matches* é representada por D_m. Esse valor é obtido da seguinte forma:

$$D_m = d | \forall m_i \in T, d_i \in m_i \wedge d_i \geq d \tag{4.1}$$

O filtro aplicado eliminou todos os *matches* cuja distância euclidiana fosse maior do que $6D_m$, ou seja, o conjunto de *matches* M após

o filtro, é definido como segue:

$$M = \{m_i | m_i \in T, d \in m_i \wedge d \leq 6D_m\} \tag{4.2}$$

A aplicação do filtro acima descrito resultou no conjunto de *matches* apresentado na figura 16.

Figura 14 – Fotos de um ambiente de escritório, utilizadas nos testes dos filtros desenvolvidos

Figura 15 – Matching entre as duas fotos apresentadas na figura 14

4.4 COERÊNCIA DE ROTAÇÃO

Cada *keypoint* SIFT possui uma orientação predominante, da variação do gradiente dos *pixels* vizinhos. Considera-se como rotação,

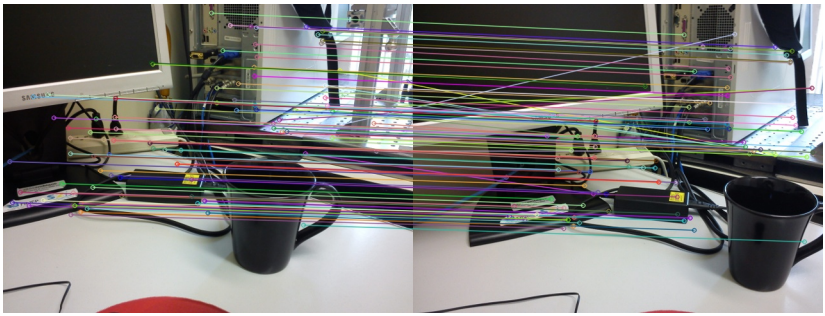


Figura 16 – Resultado da filtragem preliminar dos *matches* da figura 15, baseada apenas na menor distância euclidiana entre os *matches*

a diferença entre a orientação dos *keypoints* (k_1 e k_2) presentes em cada *match*. Como a orientação não depende de outro parâmetro, o algoritmo para filtragem de falsos *matches* pode ser implementado de uma forma mais eficiente do que o algoritmo baseado na coerência de translação. O algoritmo é dividido em três etapas: a) inicialização do vetor de frequências, b) contagem da distribuição de frequências e c) eliminação de falsos *matches*.

Primeiramente, é necessário inicializar a tabela de distribuição de frequências, dividindo-a em *bins*. Cada *bin* corresponde a uma faixa de frequências. Esta etapa está representada no algoritmo 2.

Algoritmo 2: Inicialização do vetor de frequências

Entrada:
a) prec: precisão da distribuição de frequências;
b) Match[ ] matchesInicial: Conjunto de todos os matches obtidos.
Saída: Contagem[ ] vetor_cont: Conjunto de rotações, com zero ocorrências.

```
1  início
2      vetor_cont = ∅;
3      nr_bins = 360/prec;
4      angulo_atual = 0;
5      para i=1 até nr_bins faça
6          contagem.valor_inicial = angulo_atual;
7          angulo_atual = angulo_atual + prec;
8          contagem.valor_final = angulo_atual;
9          contagem.nr_matches = 0;
10         adiciona(contagem, i, vetor_cont);
11     fim
12     retorna vetor_cont;
13 fim
```

O parâmetro *prec* define a precisão com que a contagem das frequências é realizada. Como pode ser visto no algoritmo 2, quanto menor o valor desse parâmetro, maior o número de intervalos de frequência. O intervalo próximo a 360° pode ser omitido da contagem. No entanto, para fins didáticos, ele foi mantido no algoritmo de inicialização do vetor. O conjunto de *matches* utilizado como entrada, *matchesInicial*, pode ser o conjunto de todos os *matches* ou, preferencialmente, o conjunto de *matches* resultante da eliminação de *matches* excedentes (algoritmo 1).

O vetor que armazena as contagens de frequências é formado por elementos do tipo "contagem", que possui três atributos:

- valor_inicial: ângulo inicial da faixa de rotação;
- valor_final: ângulo final da faixa de rotação;
- nr_matches: quantidade de *matches* contabilizados na respectiva faixa de rotação.

Ao final, o algoritmo retorna um vetor contendo todas as faixas de frequência, cada qual com uma contagem de zero ocorrências na respectiva frequência.

Após a inicialização do vetor de frequências, realiza-se a contagem das rotações encontradas nos *matches*. Essa etapa está representada no algoritmo 3.

Algoritmo 3: Contagem das rotações encontradas

Entrada:
a) Match[ ] matches: Conjunto de todos os matches;
b) prec: precisão da distribuição de frequências.
Saída: Contagem[ ] vetor_cont: Conjunto de rotações, com as respectivas ocorrências.

```
1 início
2 |  para i=1 até nr_matches faça
3 |  |  ind = ⌈(angf/prec)⌉;
4 |  |  vetor_cont[ind].nr_matches++;
5 |  fim
6 |  retorna vetor_cont;
7 fim
```

O algoritmo 3, contabiliza a distribuição das frequências. A variável *angf* corresponde ao valor absoluto da diferença entre os ângulos dos *keypoints* da imagem *template* (k_1) e de teste (k_2).

Com a distribuição de frequências contabilizada, passa-se à etapa de filtragem propriamente dita, realizada conforme o algoritmo 4.

Algoritmo 4: Eliminação de falsos *matches* baseada na rotação

Entrada:
a) Match[ ] matches: Conjunto de todos os matches;
b) prec: precisão da distribuição de frequências.
Saída: Caso o filtro tenha sido aplicado, retorna o conjunto de matches após a eliminação de falsos matches, caso contrário, retorna o mesmo conjunto de matches de entrada.

```
1  início
2      perc_dif = pf / sf;
3      dif_ind = |ind_pri − ind_seg|;
4      if (perc_dif ≥ 1,3) ou (dif_ind = 1) then
5          good_matches = ∅;
6          para i=1 até nr_matches faça
7              ind_ang := ⌈(angf/prec)⌉;
8              if (ind_ang ≥ ind_pri − 1) e
               (ind_ang ≤ ind_pri + 1) then
9                  adiciona(matches[i], good_matches);
10             end
11         fim
12         retorna good_matches;
13     end
14     senão
15         retorna matches;
16     fim
17 fim
```

A determinação da variação de orientação predominante nos *matches* é feita através da análise da distribuição de frequência estatística entre as diversas variações de orientação, presente nos *matches*. Variações de orientação próximas a 360° podem ser computadas na faixa de frequências próximas a 0°.

O algoritmo 4 descarta os *matches* cuja variação de orientação de k_2 em relação a k_1 estiver fora da faixa de maior frequência, calculada no algoritmo 3. As variáveis *angf* e *prec* possuem a mesma semântica utilizada no algoritmo 3. Para evitar que pequenas variações de orientação interfiram na eficácia do algoritmo, um *match* é considerado bom se estiver contido entre $-prec$ e $+prec$ em relação à rotação predominante (ω), ou seja, um bom *match* é aquele cuja rotação (γ) obedeça à condição estabelecida pela inequação 4.3:

$$(\omega - prec) \leq \gamma \leq (\omega + prec) \tag{4.3}$$

A rotação predominante (ω) é representada no algoritmo através da variável *ind_pri*, que contém o índice do vetor de frequências, com a maior contagem de *matches*. No intuito de melhorar a robustez do filtro, utilizou-se um parâmetro de proporção de coerência, *perc_dif*. Esse parâmetro é a razão entre a maior frequência (pf) e a segunda maior frequência (sf). Caso essa razão não esteja acima de um valor estipulado (descrito a seguir), considera-se que os *matches* estão muito aleatórios e não fornecem informações suficientes para um filtro robusto. Nessa situação, pode-se ignorar o filtro, adotar algumas das estratégias indicadas a seguir ou relaxar os parâmetros de *matching*, de forma que se obtenha um maior número de *matches*. No entanto, nesse último caso, há uma maior possibilidade do surgimento de *matches* aleatórios que coincidam com os bons *matches*. No caso de o índice da segunda maior frequência (*ind_seg*) estar adjacente ao da primeira, considera-se que também é um caso em que há coerência entre os *matches* e o filtro é aplicado.

Durante os experimentos realizados, verificou-se que, em geral, valores de *perc_dif* acima de 3 podem ser considerados confiáveis. Valores muito próximos a 1 não devem ser utilizados. Nesse caso, pode-se tentar aumentar a precisão da distribuição de frequência, aumentando o número de faixas de frequência. Uma outra opção seria utilizar alguma estratégia de decisão, por exemplo, baseada em lógica fuzzy pois, a faixa de rotação mais próxima da real pode estar entre a faixa de *pf* e de *sf*.

A tabela 1 apresenta os dados resultantes da distribuição de frequência estatística das faixas de rotação dos *matches* da figura 15 (ambiente de escritório), que foram utilizados para aplicar o filtro que resultou na figura 17. Como pode ser facilmente observado, a faixa de rotação com maior frequência está entre 0° e 20°, com 92 *matches*. Essa faixa poderia também incluir a frequência que seria contabilizada para a faixa de 340° a 360°. Os *matches* que estão nas faixas de "80-100", "100-120", "200-220", "220-240", "260-280" e "300-320" são considerados falsos *matches* e serão excluídos.

Conforme comentado anteriormente, o resultado da distribuição de frequência possui ainda uma segunda utilidade. Com ele, é possível detectar a inclinação de um objeto, supondo que se esteja localizando um objeto em uma cena. A faixa de rotação (variação de orientação) indica o quanto um objeto está inclinado (rotacionado) na cena, em relação à imagem *template*. A figura 17 apresenta o resultado da

Tabela 1 – Distribuição de frequências das faixas de rotação

Faixa de rotação (°)	Frequência
$0 \sim 20$	92
$20 \sim 40$	0
$40 \sim 60$	0
$60 \sim 80$	0
$80 \sim 100$	4
$100 \sim 120$	1
$120 \sim 140$	0
$140 \sim 160$	0
$160 \sim 180$	0
$180 \sim 200$	0
$200 \sim 220$	1
$220 \sim 240$	1
$240 \sim 260$	0
$260 \sim 280$	1
$280 \sim 300$	0
$300 \sim 320$	2
$320 \sim 340$	0
$340 \sim 360$	0

aplicação do filtro por coerência de rotação aos *matches* da figura 16. O resultado indica que a cena 2 não está rotacionada em relação à cena 1. Porém, como foi utilizada uma precisão de 20° (parâmetro $prec = 20$), a rotação da cena 2 em relação à cena 1 está entre $-20°$ e $+20°$. Os 10 *matches* que foram eliminados no filtro estão ilustrados na figura 18.

4.5 COERÊNCIA DE TRANSLAÇÃO

Conforme apresentado na seção 4.3, num *matching* entre duas imagens obtemos m *matches* e cada *match* m_i é composto por dois *keypoints*, k_1 e k_2. k_1 se localiza na imagem *template* e k_2 na imagem de teste. Os termos $k_1.x, k_1.y$ e $k_2.x, k_2.y$ são os valores X,Y dos *keypoints* na imagem *template* e de teste, respectivamente, correspondentes ao *match* i. Sendo assim, a translação de cada m_i é definida como:

$$dx_i = k_2.x - k_1.x \tag{4.4}$$

Figura 17 – Resultado da aplicação do filtro por coerência de rotação nos *matches* apresentados na figura 16

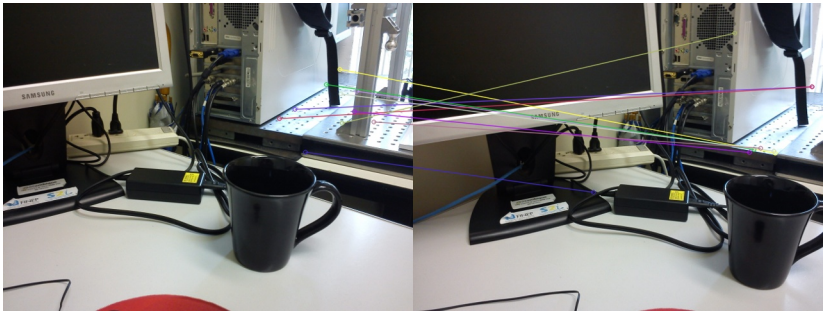


Figura 18 – *Matches* eliminados no filtro por coerência de rotação

$$dy_i = k_2.y - k_1.y \tag{4.5}$$

Nas equações 4.4 e 4.5, *dx* representa o deslocamento no eixo X e dy, o deslocamento no eixo Y, respectivamente. A forma mais simples de obter-se a translação do conjunto m (com n elementos) seria calcular a média aritmética da translação, com base em cada m_i:

$$dx = \frac{1}{n}\sum_{i=1}^{n} dx_i \tag{4.6}$$

$$dy = \frac{1}{n}\sum_{i=1}^{n} dy_i \tag{4.7}$$

No entanto, calculando a translação em X,Y com base nas equações 4.6 e 4.7 não teria-se a garantia de uma informação correta, pois, nem todos os *match* são bons (com *keypoints* corretamente localizados).

Levando em conta a coerência entre os *keypoints* de cada *match*, pode-se utilizar essa informação para eliminar os *matches* com *keypoints* não-coerentes. Como objetivo, deseja-se maximizar o número de *matches* coerentes. Inicialmente pode-se pensar em calcular a translação no eixo X, que maximiza o número de *matches* e fazer o mesmo para o eixo Y. Entretanto, o conjunto de *matches* resultante para o cálculo da translação em X e em Y pode ser mutuamente excludente.

Sejam *dx* e *dy* as translações calculadas de forma independente nos eixos X,Y, respectivamente, que maximizam o número de *matches*. Considere M_x o conjunto de *matches* resultante do filtro por translação aplicado somente no eixo X, com base na translação *dx* e M_y o conjunto de *matches* resultante do filtro por translação aplicado somente no eixo Y, com base na translação *dy*. Verifica-se que pode ocorrer a situação em que, sendo "$R = M_x \cap M_y$", "$M_x \not\subset R$" ou "$M_y \not\subset R$". Isso significa que, a translação em X que gera o maior número de *matches* pode conter *matches* que não estão presentes no conjunto de *matches* resultantes do cálculo da melhor translação em Y e vice-versa.

Com isso pode-se concluir que as translações em X e em Y são dependentes entre si, para gerar o maior número de *matches*. Dessa forma, o algoritmo para eliminar falsos *matches*, baseado na translação, procura encontrar a melhor combinação de translação em X e em Y. Para este algoritmo, considere *xm,ym* a menor largura e menor altura, respectivamente, entre as duas imagens e *tx,ty* a tolerância de deslocamento nos eixos x e y, respectivamente, para considerar que dois pontos estão na mesma posição.

Algoritmo 5: Eliminação de falsos *matches* baseada na translação

Entrada:
a) Match[ ] matches: Conjunto de todos os matches;
b) tx,ty: Tolerância no deslocamento nos eixos X e Y, respectivamente.
Saída:
a) dx_max_pontos,dy_max_pontos: Translação obtida nos eixos X e Y, respectivamente;
b) Match[ ] good_matches: Conjunto de matches após a eliminação de falsos matches.

```
1  início
2      nr_max_pontos = 0;
3      menor_dif = ∞;
4      good_matches = ∅;
5      para dx=−xm até xm faça
6          para dy=−ym até ym faça
7              nr_pontos_atual = 0;
8              somaDif = 0;
9              matches_temp = ∅;
10             para i=1 até nr_matches faça
11                 difX = |k1.x + dx − k2.x|;
12                 difY = |k1.y + dy − k2.y|;
13                 if (difX ≤ tx) and (difY ≤ ty) then
14                     nr_pontos_atual++;
15                     somaDif = somaDif + difX + difY;
16                     adiciona(matches[i], matches_temp);
17                 end
18             fim
19             if (nr_pontos_atual > nr_max_pontos) ou
               ((nr_pontos_atual = nr_max_pontos) e (somaDif
               < menor_dif)) then
20                 menor_dif = somaDif;
21                 nr_max_pontos = nr_pontos_atual;
22                 dx_max_pontos = dx;
23                 dy_max_pontos = dy;
24                 good_matches = matches_temp;
25             end
26         fim
27     fim
28     retorna dx_max_pontos,dy_max_pontos,good_matches;
29 fim
```

O algoritmo 5 encontra a translação *dx,dy* que resulta no maior número de *matches* e descarta os *matches* cuja posição dos *keypoints* da imagem de teste (k_2) estiverem fora da translação predominante (*dx_max_pontos* e *dy_max_pontos*). Para cada possível deslocamento *dx* (variando de $-xm$ a $+xm$) e *dy* (variando de $-ym$ a $+ym$), deve-se percorrer todos os *matches* e verificar se os respectivos *keypoints* k_2 estão dentro da tolerância *tx* e *ty*. Na ocorrência de se ter mais do que um par *dx,dy* que gere o número máximo de *matches*, deverá ser mantido o par que apresentar o menor valor da norma 1, ou seja, o menor valor absoluto da soma (*somaDif*). Como resultado, são eliminados os *matches* cuja posição dos k_2 estiverem fora da margem de tolerância *tx,ty*, utilizando a translação *dx,dy*. A margem de tolerância utilizada nos experimentos foi de 5%. A figura 19 apresenta o resultado da aplicação do filtro por coerência de translação nos *matches* da figura 16. A translação predominante (em *pixels*) detectada foi *dx*=-102,*dy*=-35. Isso significa que a imagem 2 está deslocada 102 *pixels* no eixo X e 35 *pixels* no eixo Y, em relação à imagem 1. Os 12 *matches* que foram eliminados no filtro estão ilustrados na figura 20.

Figura 19 – Resultado da aplicação do filtro por coerência de translação nos *matches* apresentados na figura 16

4.6 PRECISÃO E LIMITAÇÕES

As faixas de distribuição de frequência apresentadas na tabela 1 foram alocadas a cada 20°. Esse parâmetro pode ser facilmente alterado, sem grande impacto no desempenho do algoritmo. Quanto menor a faixa, mais preciso o resultado final, no entanto, se estará mais sujeito

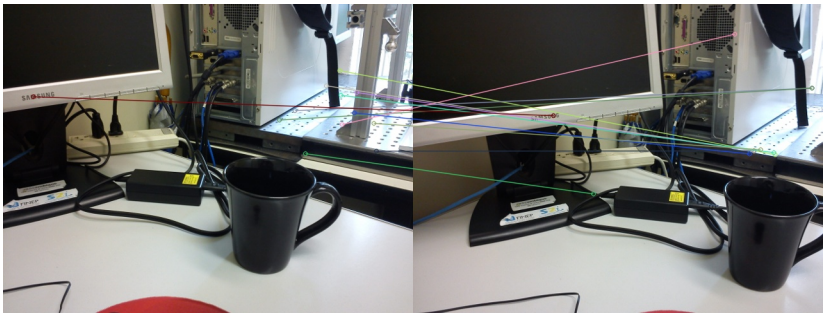


Figura 20 – *Matches* eliminados no filtro por coerência de translação

à eliminação de *matches* possivelmente bons.

Para se ter um bom resultado final, é importante que se tenha uma quantidade razoável de *matches*. Principalmente em imagens com baixo contraste, é importante que se tenha um elevado número de *matches*, reduzindo a possibilidade da coincidência de falsos *matches* com características semelhantes.

A principal limitação da abordagem proposta para eliminação de falsos *matches* é que, caso todos os *matches* ou a imensa maioria deles, sejam falsos, o algoritmo de filtragem por translação não irá detectar a situação e identificará uma translação que melhor se adequar aos *matches*. No caso da filtragem por rotação, o valor de $perc\_dif$, descrito na seção 4.4, possuirá um valor baixo e a filtragem não será possível.

Para contornar a limitação da filtragem por translação mencionada acima, pode-se verificar se a quantidade de *matches* que se encaixam no deslocamento obtido é suficientemente grande para uma filtragem segura. A filtragem por translação possui ainda outras três limitações:

- Não funciona adequadamente para *matching* de imagens em escalas muito diferentes;
- Caso a cena possua objetos com profundidades muito diferentes, o algoritmo poderá sofrer interferência. No entanto, como pode ser observado na figura 19, pequenas variações de profundidade dos objetos na cena não afetam a eficácia do algoritmo;
- Poderá não funcionar adequadamente para *matching* de imagens em que apenas uma estiver rotacionada. O funcionamento dependerá da posição dos *keypoints*, sendo que, os *keypoints* localizados

próximos ao centro de rotação terão um deslocamento menor do que aqueles localizados mais distantes. Sendo assim, o algoritmo não é recomendado nessa situação.

A filtragem por rotação possui um funcionamento semelhante a uma clusterização, já a filtragem por translação, segue uma abordagem um pouco diferente. Entretanto, percebe-se que os resultados das duas abordagens convergem. Com essa abordagem utilizada na filtragem por translação, pretende-se incluir no conjunto de bons *matches* aqueles que possivelmente estariam de fora dos *clusters*, mas que ainda assim possam ser bons *matches*, dentro de uma margem de tolerância. Nos testes realizados, percebeu-se que a filtragem por translação tende a resultar em um maior número de bons *matches*. Esse também é um dos motivos pelos quais aplica-se os dois filtros separadamente.

Para demonstrar a aplicabilidade dos algoritmos de filtragem no contexto da inspeção de PCI, foram realizados experimentos com a utilização de fotos reais de componentes eletrônicos, de forma de semelhante aos experimentos feitos com as fotos do ambiente de escritório. A figura 21 apresenta o resultado do *matching* de fotos reais de dois componentes eletrônicos do mesmo tipo. A aplicação dos filtros por coerência de rotação e translação resultou nos *matches* apresentados nas figuras 22 e 23, respectivamente. Os *matches* eliminados pelos respectivos filtros são apresentados nas figuras 24 e 25.

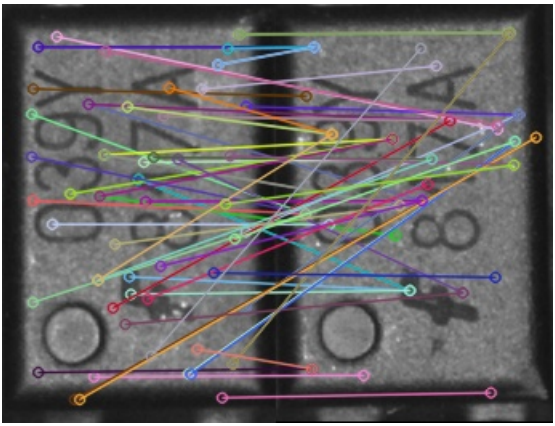


Figura 21 – *Matching* de fotos reais de dois componentes eletrônicos do mesmo tipo

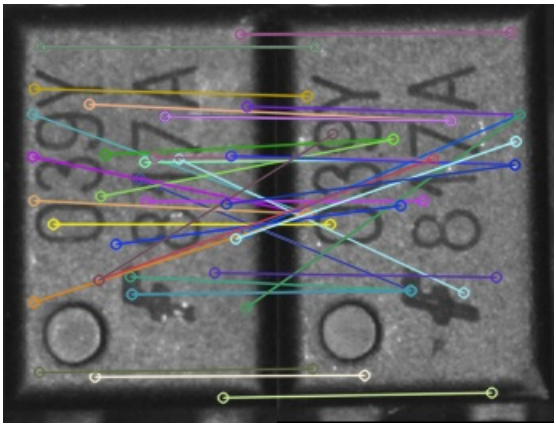


Figura 22 – Resultado da aplicação do filtro por coerência de rotação nos *matches* apresentados na figura 21

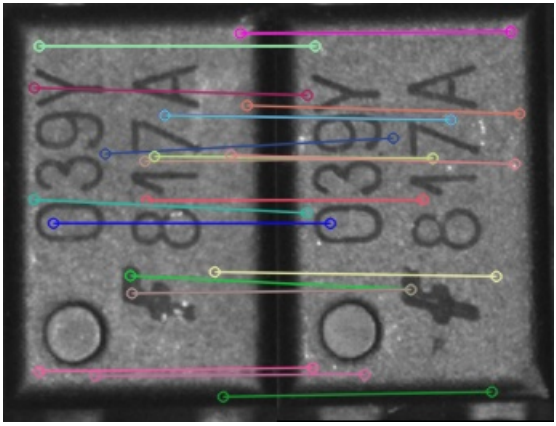


Figura 23 – Resultado da aplicação do filtro por coerência de translação nos *matches* apresentados na figura 21

A filtragem por coerência de rotação eliminou 28 *matches*, enquanto que na filtragem por coerência de translação foram eliminados 44 *matches*. Percebe-se que, apesar do filtro por coerência de translação ter eliminado uma quantidade maior de *matches*, houve convergência entre os *matches* eliminados com ambos os algoritmos.

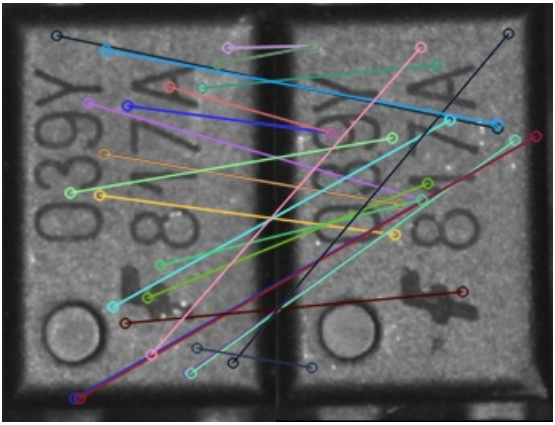


Figura 24 – *Matches* eliminados na aplicação do filtro por coerência de rotação aos *matches* apresentados na figura 21

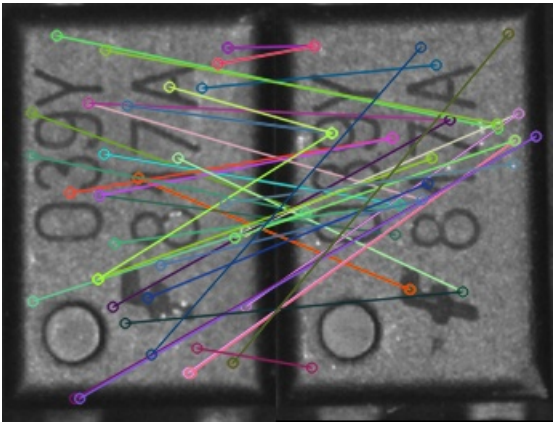


Figura 25 – *Matches* eliminados na aplicação do filtro por coerência de translação aos *matches* apresentados na figura 21

5 AVALIAÇÃO DOS RESULTADOS

No intuito de validar a arquitetura de processamento de imagens proposta, bem como, os algoritmos desenvolvidos, foram realizados diversos experimentos. As imagens utilizadas nos experimentos foram adquiridas utilizando a versão atual da S2iAOI, incluindo o software de gerenciamento da máquina e o sistema de iluminação disponível. Em função dos testes realizados para detectar a ausência de componentes, diversas fotos coloridas foram utilizadas. Essa fotos foram obtidas com o uso da mesma câmera instalada na S2iAOI, porém, com o uso do software de teste, distribuído pelo fabricante, juntamente com a câmera. Para efeito de testes, algumas dessas fotos coloridas foram também utilizadas nos experimentos relatados a seguir, porém, previamente convertidas para imagens em escala de cinza.

O sistema de iluminação da S2iAOI é composto por um *ring-light* e quatro painéis de LED. As figuras 26 e 27 apresentam fotos do sistema de iluminação atual da S2iAOI. O tipo de iluminação utilizada não foi contemplado nos experimentos realizados, de modo que, um único tipo de iluminação foi utilizado. Como, de uma maneira geral, os painéis laterais de LED proveram melhores resultados, esse tipo de iluminação foi utilizado em todos os experimentos realizados.

Dentre os componentes SMD existem variações em termos de tamanho, textura da superfície, layout da serigrafia, entre outros. Em virtude disso, diferentes PCI foram utilizadas para diversificar os testes. Todos os casos de teste realizados seguiram a mesma estrutura, descrita no capítulo 3.4.

A seguir são apresentados os resultados de 7 casos de teste, que foram criados com base em 6 modelos diferentes de PCI. Cada caso de teste representa uma inspeção. Em alguns casos, são comparadas fotos de componentes que estão no mesmo local, porém, em unidades diferentes do mesmo modelo placa. Em outros casos são utilizados componentes que pertencem à mesma placa, porém, estão em locais diferentes. Como o funcionamento de uma PCI não possui relevância na inspeção, a semântica dos circuitos testados não é apresentada.

Em todos os casos de teste foi utilizada a implementação do SIFT distribuída junto ao OpenCV 2.4.0 e os parâmetros do SIFT foram mantidos com o valor padrão, exceto os parâmetros σ da função gaussiana e threshold de contraste. O valor do parâmetro σ variou conforme o tamanho dos componentes eletrônicos e consequentemente, das dimensões das respectivas fotos. A utilização do valor padrão do σ

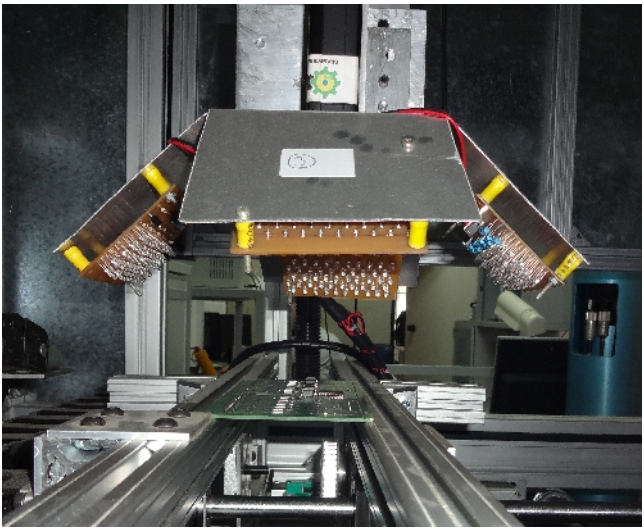


Figura 26 – Vista lateral do sistema de iluminação e câmera, utilizados na S2iAOI

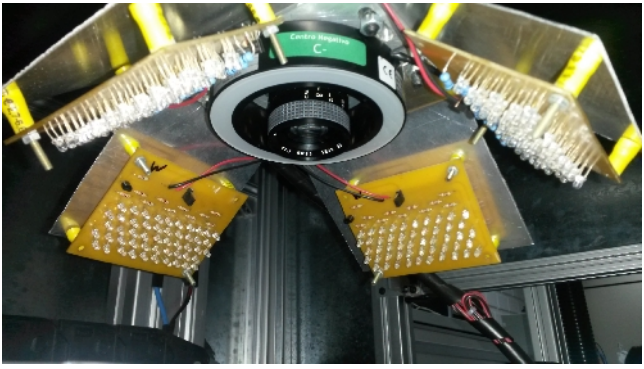


Figura 27 – Vista de baixo do sistema de iluminação e câmera, utilizados na S2iAOI

(1,6) para componentes muito pequenos tende a reduzir o número de *keypoints* gerados. Isso ocorre porque o borramento atinge uma região proporcionalmente muito grande da imagem. O oposto ocorre com fo-

tos grandes, em que, um excessivo número de *keypoints* é gerado. Além do excessivo número de *keypoints*, nesse último caso tem-se a geração de *keypoints* instáveis, visto que, são gerados *keypoints* em regiões pouco significativas das imagens. Dessa forma, verificou-se que estipulando valores fixos de σ para determinadas dimensões das fotos, acarretaria inspeções com melhores resultados. Portanto, para imagens cuja largura ou altura for inferior a 100 *pixels*, é utilizado $\sigma = 1$. Para os demais casos, é utilizado $\sigma = 3$. Esses valores foram obtidos empiricamente, analisando-se os resultados obtidos com as inspeções. O valor do parâmetro de threshold de contraste foi usado conforme sugestão de Lowe (LOWE, 2004), ou seja, 0,03, ao invés de 0,04, que é o valor padrão na implementação do SIFT. Dessa forma, há a tendência de surgir um maior número de *keypoints*, como também, um maior número de falsos *matches*. Nos experimentos realizados, o uso do valor 0,03 para esse parâmetro resultou em um considerável aumento no número de *keypoints*, enquanto que, o impacto no desempenho do sistema foi irrelevante. Por exemplo, em 40 fotos escolhidas aleatoriamente dentre as fotos de componentes eletrônicos utilizadas nos 7 casos de teste, a redução do threshold de contraste de 0,04 para 0,03 resultou, em média, num aumento de 18% na quantidade de *keypoints* gerados. Dentre essas 40 fotos, 32,5% das fotos não resultaram em aumento no número de *keypoints* e o maior aumento, para uma mesma foto, foi de 31,2% (94 *keypoints*).

Visando maximizar a quantidade de bons *matches*, nenhum filtro preliminar foi aplicado aos *matches*, após os procedimentos de matching. Um exemplo de filtro preliminar é o que foi aplicado no capítulo 4, porém, nesse caso utilizado apenas para fins didáticos. Em uma aplicação real, esse tipo de abordagem baseada em um threshold global deve ser evitada pois, alguns *keypoints* são mais distintivos do que outros (LOWE, 2004).

Ao todo foram realizados 194 testes, distribuídos em 3 situações: sem defeito, defeito real e defeito virtual. Os defeitos reais foram produzidos fisicamente nas placas, por exemplo, removendo-se um componente de uma placa de teste. Os defeitos virtuais foram produzidos editando-se as fotos obtidas para, por exemplo, simular o deslocamento de um componente através de um deslocamento na janela de inspeção. É importante observar que, como atualmente não é possível especificar janelas de inspeção na S2iAOI, todas as imagens adquiridas foram recortadas para que se pudesse gerar as janelas de inspeção necessárias.

A tabela 2 apresenta um resumo dos defeitos existentes (coluna "Ex") e encontrados (coluna "En"), em todos os casos de teste. A

tabela 3 apresenta a contabilização de defeitos que não foram reconhecidos, fotos com *keypoints* insuficientes para se realizar a inspeção, quantidade de testes em cada inspeção, duração de cada inspeção e a quantidade de falsas falhas identificadas. Todas as ocorrências de falsa falha (falso negativo) encontradas foram defeitos desconhecidos, que tratavam-se de situações com um baixo número de matches. Essas situações foram ocasionadas, em geral, pelo uso de imagens inadequadas como, por exemplo, imagens com baixo contraste ou fotos de componentes com serigrafia em má qualidade. Excetuando-se os casos de componentes com serigrafia muito semelhante, que ocasionaram falsas falhas de componentes trocados, não houve falsa falha além dos defeitos desconhecidos.

Tabela 2 – Contabilização dos resultados das inspeções para os defeitos existentes e os encontrados

Caso	**Ausentes**		**Invertidos**		**Deslocados**		**Errados**		**Corretos**		**TOTAL**	
	Ex	En	Ex	En	Ex	En	Ex	En	Ex	En	Ex	En
1	2	2	7	5	0	0	0	0	14	9	23	16
2	4	4	1	0	1	0	1	1	1	0	8	5
3	1	1	4	4	3	3	5	0	5	5	18	13
4	1	1	2	2	0	0	3	0	24	14	30	17
5	2	0	8	8	21	6	2	1	32	19	65	34
6	2	2	9	9	9	9	2	0	6	5	28	25
7	2	0	1	1	0	0	3	3	7	6	13	10
TOT	14	10	32	29	34	18	16	5	89	58	185	120
%	71%		91%		53%		31%		65%		65%	

No caso de teste 1, foi criado um teste propositalmente com uma grande variação de iluminação. Como resultado, não foram gerados *keypoints* suficientes na foto desse teste, impossibilitando a realização do mesmo. No mesmo caso de teste, também foi realizada uma modificação no contraste de uma outra foto de teste e novamente, uma quantidade insuficiente de *keypoints* foi gerada. Entretanto, em um terceiro teste também houve uma grande variação na iluminação e o teste foi bem sucedido. Esse tipo de variação na quantidade de *keypoints* pode ser verificada durante a etapa de *setup* e, caso necessário, variar a iluminação visando maximizar a quantidade de *keypoints* gerados. Para o caso de teste 1, a maior parte dos defeitos desconhecidos foi ocasionada por uma quantidade insuficiente de *matches*.

Ainda no caso de teste 1, foi realizado o matching entre um componente e uma grande área do fundo da placa, sem componente.

Tabela 3 – Resumo do desempenho e defeitos não encontrados nas inspeções

Caso	**Defeitos desconhecidos**	**Keypoints insuficientes**	**Nr. testes**	**Duração**	**Falsas falhas**
1	5	1	24	24s	4
2	3	0	8	6s	1
3	3	0	18	1min 10s	2
4	11	6	36	2min 5s	10
5	29	2	67	4min 43s	29
6	1	0	28	2min 39s	3
7	3	0	13	2s	3
TOTAL	55	9	194	11min 9s	48
%	28%	5%			

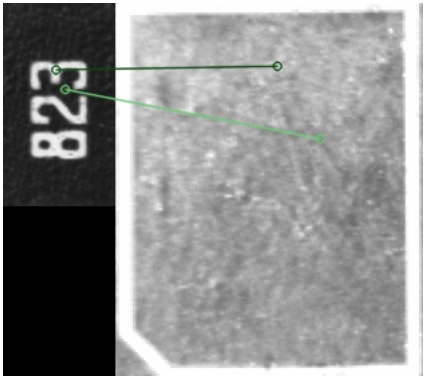


Figura 28 – Resultado do matching entre um componente e uma grande área do fundo da placa

Novamente o teste foi bem sucedido, conforme apresentado na figura 28. Isso demonstra que a arquitetura proposta é robusta o suficiente para contornar o problema do surgimento de *matches* entre *keypoints* da imagem de referência e *keypoints* originados em função de *background clutter*.

De uma maneira geral, verificou-se que uma inspeção é bem sucessida nos casos em que um razoável número de *keypoints* é gerado com as fotos dos componentes. Com isso, pode-se verificar a qualidade das fotos dos componentes no momento do *setup* da inspeção, visando

informar ao usuário sobre a robustez (ou impossibilidade) da inspeção, em função da quantidade de *keypoints* gerados com as respectivas fotos de cada componente da *golden board* (placa modelo).

Em alguns testes, os defeitos classificados como componente errado, na realidade se tratavam do mesmo componente, porém, o *match* não foi realizado com a foto do componente de referência (*gold*) mas foi realizado com a foto do mesmo componente que estava no banco de dados. Esse problema pode ser facilmente contornado verificando-se a identificação do componente que está sendo inspecionado e que foi encontrado no banco de dados. Essa situação reforça a ideia de que pode-se melhorar a robustez do sistema adicionando-se várias fotos do mesmo objeto no banco de dados. Cada uma dessas fotos de referência no banco de dados pode inclusive, ser adquirida com variações na iluminação, visando torná-la mais sucetível de ser encontrada em condições diferentes das de outras fotos do mesmo componente.

A diferença entre o total de defeitos existentes (ou componentes corretos) apresentados na tabela 2 e a quantidade de testes realizados, apresentada na tabela 3, está contabilizada na quantidade de testes com *keypoints* insuficientes (coluna "Keypoints insuficientes"), também apresentados na tabela 3.

Muitas fotos no caso de teste 4 estavam com contraste ruim ou com a serigrafia bem fraca, ocasionando o surgimento de um baixo número de *keypoints*/*matches*.

Igualmente ao caso de teste 4, no caso de teste 5, foram utilizadas propositalmente imagens com baixo contraste ou serigrafia fraca. Nesse caso de teste, também foram produzidos defeitos de rotação da foto de teste em 90°. Todos os deslocamentos não detectados no caso de teste 5 se referem a rotações de 90° aplicadas nas fotos de teste. Dentre todos os testes, apenas 1 rotação de 90° foi detectada como um deslocamento. A baixa detecção de componentes corretos foi ocasionada pela utilização de fotos que geraram poucos *keypoints* e consequentemente, poucos *matches*. As rotações de 90° fizeram com que parte dos respectivos componentes ficasse fora da foto, reduzindo consideravelmente a quantidade de *keypoints*. Essas rotações também dificultaram a aplicação do filtro por coerência de translação, uma vez que esse filtro não é indicado para ser aplicado a *matches* de imagens rotacionadas. Como a rotação aplicada ficou fora da faixa de rotação que indicaria um componente invertido, o defeito também não foi classificado como componente invertido. Para exemplificar o que é considerado um baixo número de *keypoints*, em um dos testes do caso 5, foram obtidos 7 *keypoints* com a foto de referência e 5 *keypoints* com a foto de teste.

A partir da tabela 2 percebe-se que o defeito encontrado com mais efetividade é o de componente invertido, com 91% de acertos. Em seguida, tem-se a detecção de componente ausente, com 71% de acertos. Desconsiderando-se os casos de teste 4 e 5, em que muitas imagens impróprias foram utilizadas, tem-se 92% de acertos para o defeito de componente deslocado e 76% para a situação em que não há defeito. A taxa de acertos para o defeito de componente errado (trocado) foi de 31%. A taxa média de acertos para todos os casos foi de 65%. Excluindo-se os casos 4 e 5, essa taxa sobe para 77%. É importante ressaltar que esse percentual de acertos corresponde a defeitos e componentes certos, corretamente detectados e classificados.

De um modo geral, a taxa de acertos pode ser aumentada realizando-se uma verificação da quantidade de *keypoints* gerados com cada foto de referência dos componentes. Quanto maior a quantidade de *keypoints*, maior a probabilidade de ocorrência de *matches* e quanto maior a quantidade de *matches*, menor a probabilidade de ocorrência de defeitos desconhecidos. Conforme apresentado na tabela 3, aproximadamente 28% dos defeitos (ou componentes corretos) não foram reconhecidos e 5% dos testes utilizaram imagens com uma quantidade insuficiente de *keypoints*. Com o uso de fotos adequadas e com as verificações necessárias, a taxa de acertos pode ficar acima de 80%.

Em todos os casos de teste em que havia um componente trocado e que o componente errado possuia parte da serigrafia semelhante a do componente de referência, o sistema não conseguiu detectar essa diferença. Isso é uma limitação do sistema que, acredita-se poder ser resolvida através do uso de alguma técnica de inspeção da serigrafia como, por exemplo OCV, conforme explicado no capítulo 3.2. A imensa maioria dos defeitos de componentes trocados não foram reconhecidos por esse motivo. A figura 29 ilustra um exemplo em que essa situação ocorre. Nesse exemplo, o sistema concluiu que o componente estava deslocado. Apesar de nesse teste ambos os componentes se tratarem de um "PIC 16F876A", há uma diferença na numeração abaixo da descrição do componente.

Uma situação muito semelhante à situação acima é o caso em que parte da serigrafia se repete no próprio componente. Nesses casos, percebeu-se que, especialmente nos testes com uma quantidade de *matches* relativamente baixa, após os filtros de falsos *matches*, restavam apenas os *matches* feitos entre as partes corretas da serigrafia. Principalmente o filtro por coerência de translação contribui para esse resultado. Isso se deve ao fato de este algoritmo priorizar os *matches* com a menor diferença na translação. A figura 30 ilustra um exem-

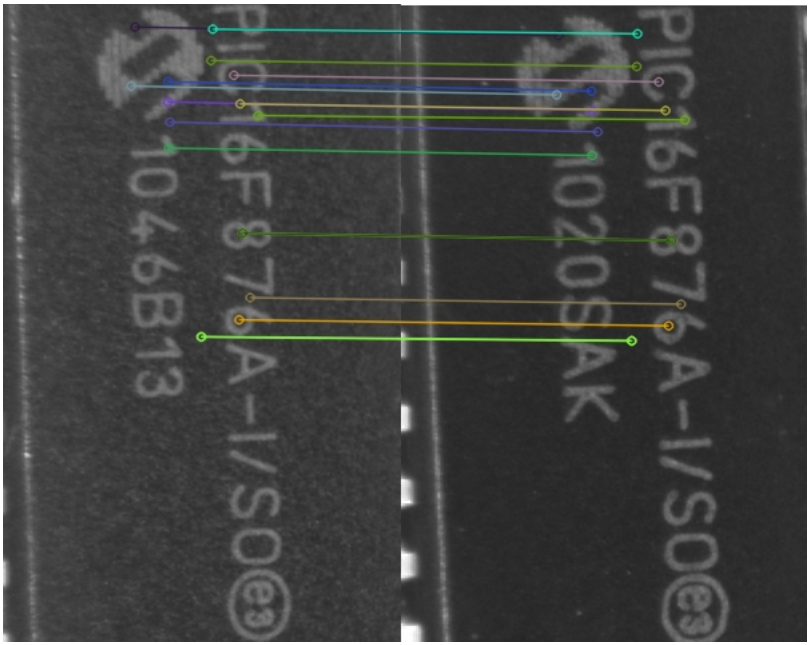


Figura 29 – Componentes reconhecidos como sendo os mesmos, em função de boa parte da serigrafia ser semelhante.

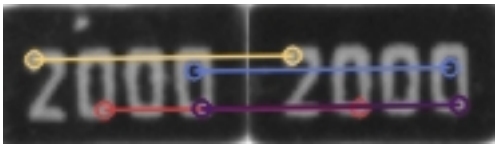


Figura 30 – Matching entre fotos de componentes com partes repetidas na serigrafia

plo dessa situação. Nessa figura observa-se a repetição do caractere "0". No entanto, como resultado final dos filtros aplicados, restou uma quantidade de *matches* suficientes para reconhecer que o componente estava correto.

Em termos de desempenho, a média geral foi de 3,5 segundos por teste. Na inspeção mais rápida obteve-se uma média de 0,15 segundos por teste. O tempo de uma inspeção depende diretamente dos

defeitos que são encontrados. Por exemplo, não havendo necessidade de se procurar por um componente no banco de dados, a inspeção será consideravelmente mais rápida. Um dos fatores que contribui significativamente para isso é o uso do filtro por coerência de translação. Esse filtro é a rotina mais custosa computacionalmente. Apesar disso, o tempo de processamento pode ser consideravelmente reduzido. Para facilitar o monitoramento, o sistema gera uma grande quantidade de imagens e mensagens de *log*, sem as quais, as inspeções seriam mais rápidas. Outras melhorias que poderiam ser feitas para aumentar o desempenho do sistema, sem nenhum impacto na eficácia, são:

- Banco de dados baseado em *keypoints*: atualmente o sistema utiliza um banco de dados com fotos dos componentes e os respectivos *keypoints* precisam ser gerados durante as inspeções. O banco de dados de *keypoints* pode ser gerado durante a etapa de *setup*;

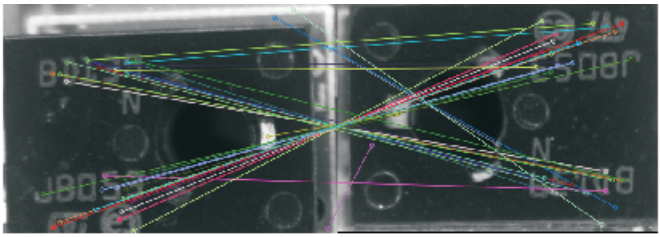
- Análise prévia do fundo da placa: para melhorar a robustez da detecção de componentes ausentes, o sistema realiza uma comparação entre o *pixel* predominante no fundo da placa e o *pixel* predominante na imagem de teste. Igualmente ao banco de dados de componentes, o cálculo do *pixel* predominante no fundo da placa pode ser realizado durante o *setup*;

- Extração prévia de *keypoints*: os *keypoints* das fotos de referência dos componentes são extraídos durante as inspeções. No entanto, essa extração pode ser feita também durante o *setup*.

Todas as melhorias acima citadas podem ser feitas automaticamente pelo sistema, sem prejudicar o processo de *setup*.

Mesmo para componentes não-SMD, percebeu-se que é possível realizar inspeções confiáveis. Dois exemplos são apresentados na figura 31. No teste "A" foram utilizadas fotos de um regulador de tensão. Como pode ser observado pelos *matches* na figura 31-A, o sistema detectou corretamente o componente invertido. No teste "B" foram utilizadas fotos de um relé. Assim como no teste "A", o defeito também foi detectado corretamente, porém, no teste "B" o componente está deslocado. Através desses testes nota-se que a arquitetura proposta é viável também para inspecionar componentes THT, desde que a aparência dos componentes, ou seja, a serigrafia, possua caracterísiticas suficientes para torná-la distintiva.

Um outro tipo de situação que pode ocorrer é a existência de superfícies de componentes muito reflexivas. Através dos experimentos realizados, verificou-se que a inspeção desses tipos de componentes é



A

B

Figura 31 – Resultado do matching utilizando fotos de componentes não-SMD

possível de se realizar, porém, exige um cuidado adicional com o sistema de iluminação. A figura 32 apresenta o resultado de três testes realizados com um cristal oscilador, cuja superfície é mais reflexiva do que a dos demais componentes utilizados. O resultado dos testes "A", "B" e "C" foram, respectivamente: sem defeito, componente deslocado e defeito desconhecido. O tipo de superfície desses componentes acarreta variações não-lineares nos *pixels*, consequentemente ocasionando divergências nos descritores dos *keypoints*. Conforme comentado no capítulo 2.6, o SIFT é apenas parcialmente invariante a esses tipos de variação.

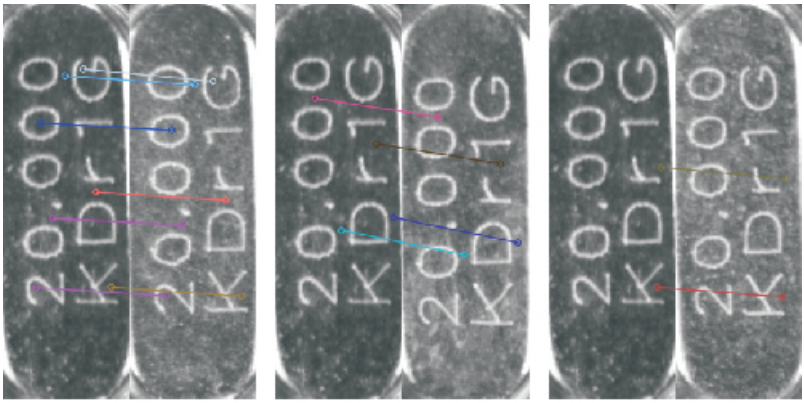


Figura 32 – Resultado do matching utilizando fotos de componentes com superfície reflexiva

6 CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvido um software de processamento de imagens com a finalidade de ser utilizado em um sistema de inspeção automática de PCI. O software desenvolvido, baseado na arquitetura de processamento de imagens proposta neste trabalho, apresentou bons resultados, comprovando que a arquitetura proposta é ao mesmo tempo, robusta e adequada à PPS.

Utilizado como elemento central da arquitetura de processamento de imagens desenvolvida neste trabalho, o algoritmo SIFT mostrou-se adequado na resolução das dificuldades inerentes à inspeção óptica automática em PPS.

Os experimentos realizados atestaram que o software desenvolvido, com as devidas melhorias necessárias, está apto a ser utilizado em um ambiente real. Com o uso do software desenvolvido, estima-se que o tempo de setup da S2iAOI, para uma placa com baixa densidade de componentes, ficaria entre 20 e 30 minutos.

A pesquisa realizada nesta dissertação fez parte de um trabalho que originou uma publicação em evento internacional, no qual foi apresentada a S2iAOI (STEMMER et al., 2013).

Muitos algoritmos de processamento de imagens apresentam bons resultados em um ambiente virtual ou altamente controlado. Como o presente trabalho visou a utilização do SIFT em inspeções reais, foi necessária uma avaliação prévia do algoritmo, no que se refere à sua aplicabilidade em um ambiente industrial. Essa avaliação originou uma segunda publicação científica, em evento internacional (LOCH; SZYMANSKI; STEMMER, 2013).

A principal vantagem da abordagem proposta é que, diferentemente de boa parte dos sistemas de inspeção automática, não requer uma fase de treinamento. Basicamente, a única configuração que necessita ser feita na etapa de *setup* é a definição de uma janela de inspeção para cada componente sendo inspecionado. Além do mais, a definição da janela de inspeção possivelmente pode ser suprimida com a utilização do arquivo CAD, contendo a definição da posição de cada componente na placa. Adicionalmente, na etapa de *setup* pode ser também especificado o tipo de defeito que se deseja detectar em cada janela de inspeção.

A principal limitação da arquitetura implementada está relacionada à detecção de componentes trocados (errados). Em função de basear-se na aparência das imagens, o SIFT torna-se, em muitos casos,

inadequado na comparação de componentes diferentes, porém, altamente semelhantes na aparência. Isso torna necessária a utilização de alguma técnica adicional, conforme comentado no capítulo 3.

É importante ressaltar que, apesar de estar voltado a contribuir com a solução do problema da inspeção automática de PCI, neste trabalho foram apresentados dois algoritmos para filtragem de falsos *matches*. A aplicabilidade desses algoritmos estende-se além da inspeção de PCI, podendo ser aplicados para filtragem de falsos *matches* em outros contextos.

De uma maneira geral, a abordagem desenvolvida buscou equilibrar a realização de inspeções robustas, com uma baixa ocorrência de falsas falhas (falsos negativos). Desconsiderando-se os casos em que foram utilizadas propositalmente imagens inadequadas, não houve ocorrência de falsa falha para os defeitos de componente ausente, invertido e deslocado. Os casos em que ocorreram falsa falha para o defeito de componente trocado, tratavam-se de componentes com serigrafia muito semelhante.

Em levantamento realizado junto ao LABelectron no ano de 2012, verificou-se que a AOI estava permitindo passar muitos erros (falsos positivos) e por isso, estava sendo considerada o gargalo da linha de produção. Em alguns casos, o percentual de erros não detectados em um lote chegava próximo a 50%. Os erros não detectados mais comuns foram: curto circuito, componente invertido e componente deslocado. Conforme apresentado no capítulo 5, esses dois últimos tipos de defeito foram detectados pelo sistema desenvolvido nesta dissertação, com 91% e 92% de confiabilidade, respectivamente.

6.1 PERSPECTIVAS PARA TRABALHOS FUTUROS

Durante o desenvolvimento desta dissertação, foram observados diversos aspectos que poderiam ser mais aprofundados, no sentido de expandir este trabalho:

- Pelo fato de estar baseada no uso do SIFT, a arquitetura proposta não é adequada para inspecionar componentes sem serigrafia. Para esse fim, sugere-se a utilização de outras técnicas, a exemplo do trabalho apresentado em (WU et al., 2010) e (WU; ZHANG; HONG, 2009);

- Visando detectar o defeito de componente ausente, a arquitetura proposta possui uma sub-rotina para verificar se determinada foto

se refere à imagem de algum componente ou a um segmento do fundo da placa. Apesar de implementada com bons resultados, diversas outras técnicas foram estudadas no sentido de tornar essa sub-rotina mais robusta. Dentre essas técnicas, foram avaliadas as Haralick *features* e técnicas baseadas em análise de histograma. Dentre as Haralick *features*, foram avaliadas 11 das 14 *features*. Sendo assim, sugere-se um estudo sobre a aplicabilidade das 3 *features* restantes. Em relação às técnicas de análise de histograma, sugere-se uma avaliação sobre variações das técnicas testadas;

- O software implementado não utilizada a identificação do componente na buscas realizadas no banco de dados de componentes. Em uma busca no banco de dados, a identificação do componente que foi encontrado pode ser utilizada para saber se trata-se do mesmo do componente que está sendo inspecionado. Isso contornará o problema dos *matches* feitos entre o componente de teste e a foto encontrada no banco de dados e que não puderam ser feitos com a imagem de referência do mesmo componente. Atualmente, o software considera esses casos como componente errado, pois não utiliza a identificação do componente;

- A abordagem proposta neste trabalho não inclui a utilização do arquivo CAD do projeto da PCI. No entanto, sugere-se uma avaliação sobre a utilização desse arquivo CAD, contendo a posição de cada componente na placa, para que o usuário não precise definir manualmente a janela de inspeção;

- Em função de apresentar um alto custo computacional, sugere-se um estudo na tentativa de melhorar o desempenho do algoritmo de filtragem de falsos *matches* por translação, apresentado no capítulo 4.5;

- Tendo em vista a relevância do algoritmo SURF, bem como, a semelhança entre as estruturas de dados dos *SIFT-keypoints* e *SURF-keypoints*, implementadas no OpenCV, sugere-se um estudo sobre o uso dos algoritmos para filtragem de falsos *matches* (apresentados no capítulo 4), baseados em *keypoints* gerados pelo algoritmo SURF;

- Pelo fato de o ambiente ao qual se destina o software implementado nesta dissertação ser um ambiente controlado, a arquitetura proposta não possui uma fase de pré-processamento das imagens. No entanto, sabe-se que, num ambiente real, variações

podem surgir, comprometendo a robustez do sistema. Mesmo que o ambiente interno da S2iAOI seja isolado contra variações na iluminação, outros tipo de variações podem surgir como, por exemplo, distorções nas imagens adquiridas. O trabalho apresentado em (LEE; PARK, 2009) propoe um método de correção de distorções geométricas para imagens. O método proposto trata de distorções arbitrárias, que não são bem resolvidas por outros métodos, os quais baseiam-se em correções utilizando modelos matemáticos, para tipos conhecidos de distorções. O método dos autores é um pouco menos eficiente para corrigir distorções conhecidas, se comparado com métodos específicos para resolver essas distorções. Porém, o método apresentado é estável para resolver genericamente os problemas de distorções. Dessa forma, sugere-se um estudo sobre as variações que podem ocorrer no ambiente interno da S2iAOI, com potencial para interferir nas inspeções. Muitas das possíveis variações que possam surgir na fase de aquisição das fotos das placas, podem ser minimizadas, a exemplo do trabalho descrito em (LEE; PARK, 2009);

- Visando possibilitar a melhoria do processo de montagem de PCI, os defeitos detectados, relacionados aos respectivos componentes, poderiam ser armazenados em banco de dados;

- O sistema de inspeção desenvolvido está focado nos defeitos gerados pelo processo de inserção de componentes. No entanto, muitos defeitos são ocasionados por problemas na etapa de soldagem. Sendo assim, sugere-se a implementação de algoritmos destinados a detectar esses tipos de defeito.

REFERÊNCIAS

BALLARD, D. H. Generalizing the Hough transform to detect arbitrary patterns. *Pattern Recognition*, v. 13, n. 2, p. 111–122, 1981.

BAY, H. et al. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, v. 110, n. September, p. 346–359, 2008.

BEIS, J. S.; LOWE, D. G. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In: *IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.]: IEEE Comput. Soc, 1997. p. 1000–1006.

BOHLOOL, M.; TAGHANAKI, S. R. Cost-efficient Automated Visual Inspection System for small manufacturing industries based on SIFT. In: *23rd International Conference Image and Vision Computing New Zealand*. [S.l.: s.n.], 2008. p. 1–6.

BRADSKI, G.; KAEHLER, A. *Learning OpenCV: Computer Vision with the OpenCV Library*. [S.l.]: O'Reilly Media, 2008. 580 p.

BROWN, M.; LOWE, D. Invariant Features from Interest Point Groups. In: *British Machine Vision Conference - BMVC*. [S.l.: s.n.], 2002. p. 253–262.

BRUNELLI, R. *Template Matching Techniques in Computer Vision - Theory And Practice*. [S.l.: s.n.], 2009.

CHO, H.-J.; PARK, T.-H. Template matching method for SMD inspection using discrete wavelet transform. In: *SICE Annual Conference*. [S.l.]: Ieee, 2008. p. 3198–3201.

DEMIR, D. et al. Quality Inspection in PCBs and SMDs Using Computer Vision Techniques. In: *International Conference on Industrial Electronics, Control and Instrumentation*. [S.l.: s.n.], 1994. v. 2, p. 857–861.

DORO, M. M. *Sistemática para implantação da garantia da qualidade em empresas montadoras de placas de circuito impresso*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2004.

DORO, M. M. *Solução integrada para auxiliar na garantia da qualidade na produção em pequenos lotes*. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2009.

DORO, M. M. *Planejamento automático das inspeções e testes aplicado ao processo de montagem de placas eletrônicas*. [S.l.], 2013. 82 p.

FRIEDMAN, J. H.; BENTLEY, J. L.; FINKEL, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, v. 3, n. 3, p. 209–226, 1977.

GRABNER, M.; GRABNER, H.; BISCHOF, H. Fast Approximated SIFT. In: *Asian Conference on Computer Vision*. Hyderabad: Springer, 2006. p. 918–927.

GUAN, S.-a.; GUO, F. A New Image Enhancement Algorithm for PCB Defect Detection. In: *International Conference on Intelligence Science and Information Engineering*. [S.l.: s.n.], 2011. p. 454–456.

HARALICK, R.; SHANMUGAM, K.; DINSTEIN, I. Textural Features for Image Classification. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3, n. 6, p. 610 – 621, 1973.

HATA, S. et al. Assembled PCB visual inspection machine using image processor with DSP. In: *15th Annual Conference of IEEE*. [S.l.: s.n.], 1989. v. 3, p. 572 – 577.

IBRAHIM, Z.; AL-ATTAS, S.; ASPAR, Z. Analysis of the Wavelet-based Image Difference Algorithim for PCB Inspection. In: *41st SICE Annual Conference*. [S.l.: s.n.], 2002. p. 2108–2113.

IBRAHIM, Z. et al. A Noise Elimination Procedure for Printed Circuit Board Inspection System. In: *Second Asia International Conference on Modelling & Simulation*. [S.l.]: Ieee, 2008. p. 332–337.

IPC. *Renewed Growth Seen in North American Electronics Industry*. 2013. <http://www.ipc.org/ContentPage.aspx?pageid=Current-Industry-Trends>.

IPC. *World PCB Market Grew 1.7 Percent in 2012 According to IPC World PCB Production Report*. 2013. <http://www.ipc.org/ContentPage.aspx?pageid=World-PCB-Market-Grew-in-2012>.

JADHAV, S. A. *Setup approval and self starting schemes for short production runs.* Dissertação (Mestrado) — Wichita State University, 2005.

JAIN, R.; KASTURI, R.; SCHUNCK, B. G. *Machine Vision.* [S.l.]: McGraw-Hill, 1995. 549 p.

JURIE, F.; DHOME, M. Real Time Robust Template Matching. In: *British Machine Vision Conference.* [S.l.: s.n.], 2002. p. 123–132.

KE, Y.; SUKTHANKAR, R. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* [S.l.: s.n.], 2004. p. II–506 – II–513 Vol.2.

KHAN, N. Y.; MCCANE, B.; WYVILL, G. SIFT and SURF Performance Evaluation Against Various Image Deformations on Benchmark Dataset. In: *International Conference on Digital Image Computing: Techniques and Applications.* [S.l.: s.n.], 2011. p. 501–506.

KIM, N.-h.; PYUN, J.-y. REAL-TIME INSPECTION SYSTEM FOR PRINTED CIRCUIT BOARDS. In: *IEEE International Symposium on Industrial Electronics.* [S.l.: s.n.], 2001. v. 1, p. 166–170.

KUMAR, G. K.; PRASAD, G. R.; MAMATHA, G. Automatic object searching system based on Real Time SIFT Algorithm. In: *IEEE International Conference on Communication Control and Computing Technologies.* [S.l.: s.n.], 2010. p. 617–622.

LAGANIèRE, R. *OpenCV 2 Computer Vision Application Programming Cookbook.* [S.l.: s.n.], 2011. 287 p.

LEE, W.-y.; PARK, T.-h. Correction Method for Geometric Image Distortion with Application to Printed Circuit Board Inspection Systems. In: *ICROS-SICE International Joint Conference.* [S.l.: s.n.], 2009. v. 2, p. 4001–4006.

LEFEBVRE, F.; CZYZ, J.; MACQ, B. A robust soft hash algorithm for digital image signature. In: *International Conference on Image Processing.* [S.l.: s.n.], 2003. p. II – 495–498 vol.3.

LETA, F. R.; FELICIANO, F. F. Computational System to Detect Defects in Mounted and Bare PCB Based on Connectivity and Image Correlation. In: *15th International Conference on Systems, Signals and Image Processing.* [S.l.: s.n.], 2008. p. 331 – 334.

LETA, F. R.; FELICIANO, F. F.; MARTINS, F. P. R. Computer Vision System for Printed Circuit Board Inspection. In: *19th International Congress of Mechanical Engineering*. [S.l.: s.n.], 2007. v. 3, p. 623–632.

LI, R. Muti-Source Remote Sensing Imageries matching based on SIFT Feature with Match-Support Measurement. In: *International Symposium on Image and Data Fusion*. [S.l.: s.n.], 2011. p. 1–4.

LIN, L.-z. et al. Study of PCB Automatic Optical Inspection System Based on Mathematical Morphology. In: *International Conference on Computer Technology and Development*. [S.l.]: Ieee, 2009. p. 405–408.

LIN, S.-C.; SU, C.-H. A Visual Inspection System for Surface Mounted Devices on Printed Circuit Board. In: *International Conference on Intelligence Science and Information Engineering*. [S.l.: s.n.], 2006. p. 454 – 456.

LOCH, G. N.; SZYMANSKI, C.; STEMMER, M. R. Evaluation of SIFT in machine vision applied to industrial automation. In: *IEEE 11th International Conference on Industrial Informatics*. [S.l.: s.n.], 2013.

LOH, H.-H.; LU, M.-S. Printed Circuit Board Inspection Using Image Analysis. In: *IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS*. [S.l.: s.n.], 1999. v. 35, n. 2, p. 426–432.

LONGJIANG, Y.; YING, Y.; SHENGHE, S. A PCB Component Location Method Based on Image Hashing. In: *International Conference on Electronic Measurement and Instruments*. [S.l.: s.n.], 2007. p. 2–697 – 2–700.

LOWE, D. G. Object Recognition from Local Scale-Invariant Features. In: *International Conference on Computer Vision*. [S.l.: s.n.], 1999. p. 1–8.

LOWE, D. G. Local feature view clustering for 3D object recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.]: IEEE Comput. Soc, 2001. p. 682–688.

LOWE, D. G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, v. 60, n. 2, p. 91–110, nov. 2004.

LUCINTEL. *Global Printed Circuit Board Industry 2012-2017: Trend, Profit, and Forecast Analysis*. [S.l.], 2013.

MALAMAS, E. N. et al. A survey on industrial vision systems, applications and tools. *Image and Vision Computing*, v. 21, n. 2, p. 171–188, 2003.

MELO, D. F. F. de. *Desenvolvimento de Máquina Automática para Inspeção Óptica de Placas de Circuito Impresso em Pequenas Séries*. 123 p. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2013.

MELO, P. R. d. S.; RIOS, E. C. D.; GUTIERREZ, R. M. V. *Placas de circuito impresso: mercado atual e perspectivas*. [S.l.], 2001. 136 p.

MIKOLAJCZYK, K.; SCHMID, C. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 27, n. 10, p. 1615–1630, 2005.

MOGANTI, M. et al. Automatic PCB Inspection Algorithms : A Survey. *Computer Vision and Image Understanding*, p. 1–46, 1996.

MUJA, M.; LOWE, D. G. FAST APPROXIMATE NEAREST NEIGHBORS WITH AUTOMATIC ALGORITHM CONFIGURATION. In: *VISAPP International Conference on Computer Vision Theory and Applications*. [S.l.: s.n.], 2009.

MUJA, M.; LOWE, D. G. *FLANN - Fast Library for Approximate Nearest Neighbors*. 2013. <http://www.cs.ubc.ca/research/flann/>.

OH, H.-W.; JUNG, J.-H.; PARK, T.-H. Gerber-Character Recognition System of Auto-Teaching Program for PCB Assembly Machines. In: *SICE Annual Conference*. [S.l.: s.n.], 2004. p. 300–305.

OLIVEIRA, F. G. de; PIO, J. L. d. S. Inspeção Visual de Placas de Circuito Integrado com Alta Densidade de Microcomponentes. In: *IV Workshop de Visão Computacional*. [S.l.: s.n.], 2008. p. 1–4.

OPREA, S. et al. Digital image processing applied in determination of misplaced components on a PCB , marked with colour code. In: *30th ISSE*. [S.l.: s.n.], 2007. p. 410–413.

PFEIFER, T. et al. CAPABLE PROCESSES BY OPTICAL METROLOGY. In: *XVII IMEKO World Congress*. [S.l.: s.n.], 2003.

PFEIFER, T. et al. Optical metrology - Enabeling factor for successful production. In: *3o Congresso Brasileiro de Metrologia*. [S.l.: s.n.], 2003.

PFEIFER, T. et al. Cognitive Production Metrology : A new concept for flexibly attending the inspection requirements of small series production. In: *36th International Matador Conference*. [S.l.: s.n.], 2010. p. 359–362.

Pirzada, Syed Jahanzeb HussainBait, M. W.; HAL, E. ul; SHIN, H. A New Adaptive Threshold Technique for Improved Matching in SIFT. In: *IEEE 54th International Midwest Symposium on Circuits and Systems*. [S.l.: s.n.], 2011. v. 00.

PUTERA, S. I.; IBRAHIM, Z. Printed Circuit Board Defect Detection Using Mathematical Morphology and MAT LAB Image Processing Tools. In: *International Conference on Education Technology and Computer*. [S.l.: s.n.], 2010. v. 5, p. V5–359 – V5–363.

RAU, H. et al. Fuzzy Reasoning for PCB Inspection. In: *Eighth International Conference on Machine Learning and Cybernetics*. [S.l.: s.n.], 2009. p. 12–15.

ROLOFF, M. L. *Um sistema multiagente para produção de pequenas séries*. 142 p. Tese (Qualificação de Doutorado) — Universidade Federal de Santa Catarina, 2013.

SCHMITT, R.; PAVIM, A. Flexible optical metrology strategies for the control and quality assurance of small series production. In: LEHMANN, P. H. (Ed.). *SPIE - Optical Measurement Systems for Industrial Inspection VI*. [S.l.]: Society of Photo-Optical Instrumentation Engineers, 2009. v. 7389, p. 738902–738902–12.

SCHNEIDER, M.; CHANG, S.-F. A robust content based digital signature for image authentication. In: *International Conference on Image Processing*. [S.l.: s.n.], 1996. p. 227 – 230 vol.3.

SHIN, D. K.; MOON, Y. S. A Robust SIFT-Based Matching Algorithm for the Occluded Area and Complex Texture. In: *5th International Colloquium on Signal Processing & Its Applications*. [S.l.: s.n.], 2009. p. 130–134.

SINHA, U. *SIFT: Scale Invariant Feature Transform*. 2013. <http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/>.

STEMMER, M. R. et al. *Cognitive Production Metrology for Flexible Small Series Production. Work progress report and project proposal for the 2nd phase. CAPES Reference Number: AUXPE Bragecrim 013/09.* [S.l.], 2011. 54 p.

STEMMER, M. R. et al. A Machine Vision System for Quality Assurance in Small Series Production. In: *11th International Symposium of Measurement Technology and Intelligent Instruments.* [S.l.: s.n.], 2013.

SUNDARAJ, K. PCB Inspection for Missing or Misaligned Components using Background Subtraction. *WSEAS Transactions on Information Science and Applications*, v. 6, n. 5, p. 778–787, 2009.

TEAM, O. D. *OpenCV*. 2013. <http://opencv.org/>.

WANG, Z. Y.; LI, Y.; LUO, Z. An Automatic Chip Character Checking System for Circuit Board Quality Control. In: *29th Annual Conference of the IEEE.* [S.l.: s.n.], 2003. p. 1767 – 1770 Vol.2.

WOOLDRIDGE, M. *An Introduction to Multiagent Systems.* [S.l.]: John Wiley & Sons, 2002. 348 p.

WOOLDRIDGE, M. J. et al. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* [S.l.]: Gerhard Weiss, 1999. 643 p.

WU, H. et al. Automated Visual Inspection of Surface Mounted Chip Components. In: *International Conference on Mechatronics and Automation.* [S.l.: s.n.], 2010. p. 1789–1794.

WU, H.; ZHANG, X.-M.; HONG, S.-L. A Visual Inspection System for Surface Mounted Components Based on Color Features. In: *IEEE International Conference on Information and Automation.* [S.l.: s.n.], 2009. p. 571–576.

WU, Y.; DAI, M. Matching Feature Points and Discarding False Matching Pairs. In: *IEEE International Conference on Information Acquisition.* [S.l.: s.n.], 2005. p. 458–463.

YAN, W. et al. A Method for Recognition of Work-pieces Based on Improved SIFT Characteristics Matching. In: *World Congress on Software Engineering.* [S.l.: s.n.], 2009. v. 1, p. 29–33.

YANG, Y.; LIU, W.; ZHANG, L. Study on Improved Scale Invariant Feature Transform matching algorithm. In: *Second Asia International*

Conference on Circuits, Communications and System (PACCS). [S.l.: s.n.], 2010. v. 262, n. 1, p. 398–401.